Sim2Real With Neural Processes

Jonas Scholz

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in
Machine Learning and Machine Intelligence

Jesus College 16th of August, 2023

Abstract

Convolutional Conditional Neural Processes (ConvCNPs) are recently developed, flexible, scalable, deep-learning-based spatiotemporal models that produce well-adjusted predictions and uncertainties. Because they, unlike Gaussian Processes, need to *learn from data* how to condition on context observations to form posterior predictive distributions, they can be data-hungry to train. We show that in settings where real data are insufficient for training, simulators that generate related, but lower-quality, synthetic data can serve as stepping stones for training ConvCNPs: The ConvCNP is first pre-trained on simulator data and then fine-tuned on real data, which we call *Sim2Real* transfer, to produce a model that is significantly more powerful than the same model trained only on simulator data or only on real data. To our knowledge, this work represents the first attempt to apply fine-tuning, which has proven very useful in domains such as computer vision or natural language processing, to spatiotemporal models. We expose and investigate the challenges specifically associated with fine-tuning in this domain and propose and evaluate approaches for how to overcome them.

We qualitatively and quantitatively characterise different domain-adaptation methods and their performance in different data-availability regimes.

Decleration of Originality

I, Jonas Scholz of Jesus College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

J. Scholz 16th of August, 2023.

Decleration of Software

This project's code is published under github.com/jonas-scholz123/sim2realgp and github.com/jonas-scholz123/sim2real-downscaling. Chapters 4 and 5 build upon the following software:

1. *Neuralprocesses* (Bruinsma, 2023) for the basic model used,
2. *Deepsensor* (Andersson, 2023) for handling the interface between weather data and tensors, which we help improve (through feedback and code contribution) while working on the project,
3. *PyTorch* (Paszke et al., 2019) for modifying the *neuralprocesses* models for our experiments,
4. *Geopandas* Jordahl et al. (2020), Shapely (Gillies et al., 2007–) Cartopy (Met Office, 2010 - 2015) and XArray (Hoyer et al., 2016) for manipulation of geo-data,
5. As well as the python programming language and commonly used python packages: *Numpy, scipy, pandas, matplotlib.*

J. Scholz 16th of August, 2023.

Word Count

Excluding declarations, bibliography, and text within figure images, but including footnotes, figure captions, tables, and appendices, this report consists of

14,700 words.

Acknowledgements

I want to thank my supervisors, Tom Andersson and Richard Turner for their support, advice, insights, ideas, encouragement, and most of all patience throughout this project. They, alongside Anna Vaughan and James Requeima, made me feel instantly welcome within their group: by showing genuine interest in my project, but also by creating such a positive atmosphere that I didn't just feel comfortable to ask (countless) questions, but welcome to.

I also want to thank everyone in my life who kept me (somewhat) sane, above all Hannah Qureshi, whom I could always rely on throughout this very stressful year. I want to also thank my Mum and Dad for their constant support and much-needed phone calls that never failed to cheer me up.

I couldn't have done the year without my amazing friends either – both those that I've made here and those I've had the joy of knowing for longer. *Roost?* is such a beautiful group of unique, kind and funny people and has shaped my "Cambridge experience" more positively than anything else, I can count myself very lucky to have met them all so early on, and I will thoroughly miss having them all within a short walk of my home.

Outside of Cambridge, I want to thank my friends from back home, which by now includes both Frankfurt and London. Visiting them was something I looked forward to the most throughout the year – whenever I came back from my trips, I felt like they were always too short.

Lastly, I want to thank the hard-working staff of the Jesus College Caff and Roost, who have kept me fed (and caffeinated). Catching up with my friends there daily made these places feel like an Oasis when the workload was at its peak.

Contents

Glossary

CNN A neural network architecture that is translation equivariant. 12

ConvCNP A translation-equivariant spatio-temporal model that predicts Gaussian distributions over target variables. 8–13, 15–18, 20, 23, 25, 27, 28, 30, 34, 39, 43–46, 53, 54, 56, 57

DWD German Weather Service (**D**eutscher **W**etter**d**ienst), whose station data we use. 33, 35

ERA5 An hourly gridded dataset of weather variables that incorporates real observations. 25, 27, 30–34, 38, 44, 45, 51–53, 56

FiLM FiLM layers affinely transform entire feature maps. They are lightweight, consisting of only 2 parameters per feature map.. 16, 19, 21, 22, 24, 26–28, 38, 41, 42, 56

GP A stochastic process where any finite subset of random variables has a multivariate Gaussian distribution. 8, 9, 13, 23–25, 28, 30

MAE The average absolute error between a prediction and truth. 32, 40, 50

NLL The negative log-likelihood of a conditional distribution is a quantity that measures how unlikely the given distribution is to have generated the observed data. 39, 40, 50, 51

Sim2Real The process of using real data to improve a system relying on simulated data. 7–9, 12, 13, 18, 23, 25, 27, 29–31, 33, 38, 40–43, 47, 51–56, 58

Chapter 1

Introduction

1.1 Motivation

In many data-driven domains, the acquisition of high-quality data is difficult, expensive, slow or otherwise restricted. Robots interacting with the physical world, for example, are often too slow to gather sufficient amounts of data for downstream applications. For autonomous vehicles, the acquisition of real-world driving data can even be dangerous to surrounding people and property. In the domain we focus on in this thesis, weather modelling, sensors can be very costly to install and maintain (especially in remote regions such as Antarctica), which is also the case for measurement sensors in other domains such as telescopes, particle accelerators or satellites.

To overcome this problem, simulators are often employed to capture some aspects of the underlying data, often by leveraging expert knowledge to approximate the underlying processes. Examples include simulated "sandbox" environments in which robots or autonomous vehicles can learn without damaging themselves (or anything else), or weather simulations, which leverage our knowledge of fluid dynamics and atmospheric physics. These simulations are usually scalable and, given enough computational resources, can generate vast amounts of data.

The problem with this approach is that simulators (by definition) only approximate reality, and the vast amount of data they generate incorporate all of these approximations, so there is usually a mismatch between the simulated and real data — which we call the Sim2Real gap. A secondary issue is the domain-specific difficulty of applying

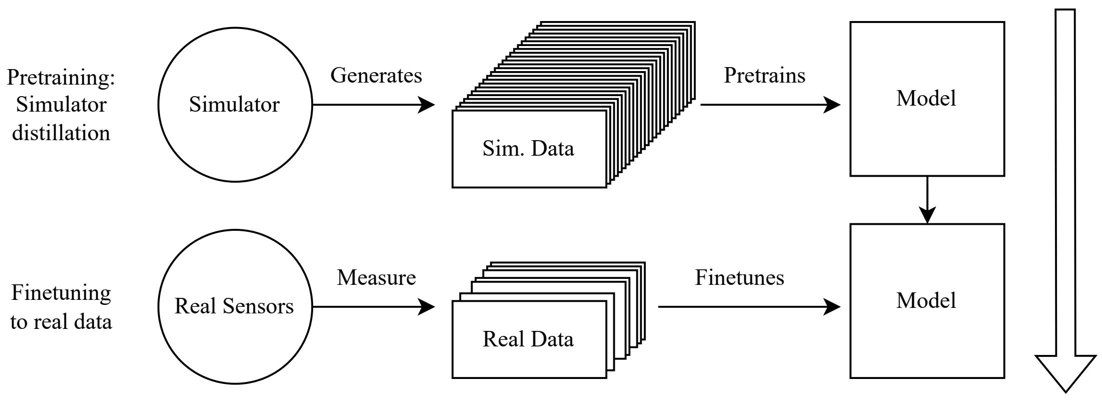


Figure 1.1: High-level Sim2Real overview. A data-hungry model is pre-trained on abundant simulator data and fine-tuned on limited real data.

a simulator to real-world scenarios: it can be challenging to apply simulators to real-world scenarios, for example, because it is challenging to map a real-world scenario to corresponding simulator parameters and initial conditions.

1.2 Sim2Real

Sim2Real transfer is the process of leveraging the (potentially small amounts of) real data to bridge the Sim2Real gap. An example of partial Sim2Real that is specific to weather modelling is *reanalysis*, where observational weather measurements are incorporated into the physics-based simulations to create a weather simulation that is consistent with observed data. This process relies on the *4d-Var* data assimilation approach (Courtier et al., 1994), which itself requires expert knowledge and is very computationally expensive (Courtier et al., 1994; Clayton et al., 2013; Hersbach et al., 2020). Additionally, the data are still forced onto a discrete grid, so a Sim2Real gap remains.

In this thesis, we focus on Sim2Real for *domain-agnostic* deep-learning models. Because of their flexibility and applicability to many domains our deep-learning approach to Sim2Real can be applied to many areas of spatio-temporal modelling.

We pre-train the deep-learning model on simulator data across a vast range of scenarios, effectively distilling the simulator into the model. We then adjust the model by fine-tuning it on *real* data to improve real-world predictions. Because the model behaviour is entirely encoded in its parameters, gradient-based optimisation can be used both for the simulator pre-training and the real data finetuning.

The type of model we focus on is the Convolutional Conditional Neural Process (ConvCNP), but we take care to address issues in a model-agnostic way wherever possible.

1.3 Sim2Real for ConvCNPs

ConvCNPs are flexible and scalable spatiotemporal deep-learning models that can make predictions at any target location given a small number of context observations, that can lie anywhere on or off the grid. At any target location, they are able to output a predicted mean and well-calibrated standard deviation for the target variable. A more detailed explanation of the ConvCNP and the particular architecture used for our experiments are shown in Sec. 2.2 and 3.2.1 respectively.

Because ConvCNPs use convolutional layers, they allow us to apply fine-tuning methods developed for Convolutional Neural Networks (e.g. for computer vision) and apply them in the novel and challenging domain of off-the-grid spatiotemporal modelling. This could open the door to developing foundation models (which have seen breakthrough successes in other domains) in completely new domains.

In contrast to other spatiotemporal models, such as Gaussian Process (GP)s, ConvCNPs enjoy computationally cheap $\mathcal{O}(N)$[1] predictions. The drawback is that they

[1] N is the total number of context observations given to the model + the number of points we want to predict at.

require a large amount of data to be trained, because they, unlike GPs, need to *learn* how to condition on data to form posterior predictive distributions.

ConvCNPs therefore lend themselves well to Sim2Real setups, where we can use vast amounts of simulator-generated scenarios to train the model, and then apply the model to real-world scenarios for cheap predictions. This method has recently been exploited successfully in toy problems such as predator-prey simulations (Gordon et al., 2019), and real problems such as determining ideal sensor locations for weather stations (Andersson et al., 2023).

One limitation of these approaches is that the models used have not bridged the Sim2Real gap – their predictions and uncertainty estimates are based on simulated, not real data. The main contribution of this thesis is investigating under what regimes of data-availability real data can be used to fine-tune simulator-pre-trained ConvCNPs in order to improve predictions on real data, and by what methods.

1.4 Weather Modelling using ConvCNPs

ConvCNPs have recently yielded promising results in weather modelling applications, where weather simulators are distilled into a ConvCNP (Vaughan et al., 2022; Andersson et al., 2023). Once trained, these models have the advantage of computationally cheap predictions that do not have to lie on a fixed grid of coordinates. We provide further details in Sec. 2.4.1.

1.5 Contributions

The goal we're working towards in this thesis is the Sim2Real transfer of a ConvCNP model trained on simulated temperature data to real temperature data – and an evaluation of various adaptation approaches in different data-availability regimes. Our findings are displayed in Sec. 5.

As a stepping stone and first experiment, we investigate Sim2Real transfer on a synthetic data set where we can control both the "simulated" and the "real" data. This allows us to test domain adaptation across a range of scenarios. Specifically, we initially train a ConvCNP to imitate one "simulation" GP and fine-tune it to then imitate a "real" GP with different properties. The relatively simple problem of 1D GP regression allows us to iterate quickly and design problems analogous to the more complicated temperature modelling problem. Results for the GP regression experiments are shown in Sec. 4.

In the proceeding sections, Sec. 2 and Sec. 3, we summarise existing relevant literature and our overarching experimental setup, respectively.

Throughout our experiments, we attempt to make our conclusions as widely transferable as possible to other models and domains.

Chapter 2

Background

This section will introduce, at a high level, relevant background information for this thesis. First, we discuss the notation employed across the whole report. We then, in more detail, summarise the Convolutional Conditional Neural Process (ConvCNP), because we use and modify it throughout this project. Finally, we provide background on weather modelling, which is the key application to which we apply our methods.

2.1 Notation

Throughout this thesis, we denote input variables (in our case spatial coordinates) as $\mathbf{x}$ and output variables as y, with

$$\mathbf{x} \in \mathbb{R}^{d_x}, \quad y \in \mathbb{R}^{d_y}. \tag{2.1}$$

To ease notation[1], we denote output variables y as scalars, but the extension to $d_y > 1$ is straightforward.

When considering N input/output variables $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, we denote them as

$$X = \begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N\end{bmatrix}^T \quad \mathbf{y} = \begin{bmatrix}y_1 & y_2 & \dots & y_N\end{bmatrix}^T. \tag{2.2}$$

The spatiotemporal models we consider predict *conditional* distributions

$$q_\theta(\mathbf{y}_T | X_T, C), \tag{2.3}$$

where target predictions $\mathbf{y}_T$ at target locations X_T are *conditioned* on context observations $C = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_C}$. Parameters θ (in conjunction with the model architecture) determine *how* predictions are conditioned and are learned via gradient-based optimisation. Throughout, we denote *target* points by a subscript T and *context* points by subscript C.

2.2 Convolutional Conditional Neural Processes

ConvCNPs (Gordon et al., 2019) are spatiotemporal models with parameters θ that define the conditional distribution as a Gaussian distribution

$$q_\theta(\mathbf{y}_T | X_T, C) = \mathcal{N}(\mathbf{y}_T; \boldsymbol{\mu}_\theta(C), \Sigma_\theta(C)). \tag{2.4}$$

[1]And because our experiments only consider scalar outputs

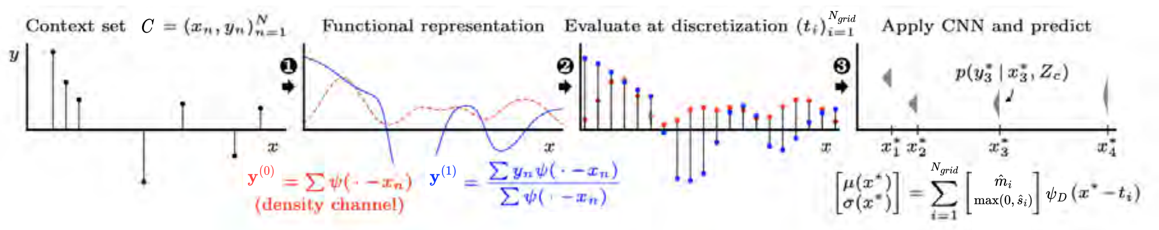

Context set $C = (x_n, y_n)_{n=1}^{N}$
Functional representation
Evaluate at discretization $(t_i)_{i=1}^{N_{grid}}$
Apply CNN and predict
y
x
x
x
❶
❷
❸
$\mathbf{y}^{(0)} = \sum \psi(\cdot - x_n)$
(density channel)
$\mathbf{y}^{(1)} = \frac{\sum y_n \psi(\cdot - x_n)}{\sum \psi(\cdot - x_n)}$
$p(y_3^* \mid x_3^*, Z_c)$
x_1^* x_2^* x_3^* x_4^*
$\begin{bmatrix} \mu(x^*) \\ \sigma(x^*) \end{bmatrix} = \sum_{i=1}^{N_{grid}} \begin{bmatrix} \hat{m}_i \\ \max(0, \hat{s}_i) \end{bmatrix} \psi_D (x^* - t_i)$

Figure 2.1: A 1-D ConvCNP. From left to right: 1: The context set input to the model. 2: The context set is encoded into a continuous function (blue) and associated density channel (red). 3: The channels are discretised onto a fixed internal grid. 4: The gridded representation is processed by a CNN and decoded into the desired output distribution. Figure taken from Gordon et al. (2019) and adapted to our notation.

Note that, in contrast to the related Convolutional *Gaussian* neural process (Markou et al., 2021a), ConvCNP models predict each target prediction y_{Ti} as conditionally independent of other predictions $y_{Tj}, i \neq j$ given the context data C i.e. $\Sigma = \text{diag}(\boldsymbol{\sigma}^{\mathbf{2}})$. Here, i, j index two distinct locations at which we compute predictions. This limitation can be alleviated through the use of auto-regressive ConvCNPs, which can model rich joint distributions over the target predictions with no modification of the model or the training procedure Bruinsma et al. (2022).

ConvCNPs compute $\boldsymbol{\mu}, \boldsymbol{\sigma}$ by encoding C onto an internal functional representation as $\mathbf{y}(C)(\cdot)$, and processing that representation into a mean function $m(\cdot)$ and standard deviation function $s(\cdot)$, using a function[2] ρ_θ:

$$m(\cdot),\ s(\cdot) = \rho_\theta(\mathbf{y}(C)). \tag{2.5}$$

These functions can finally be evaluated at target locations $\mathbf{x}_{Tj}$ to obtain predictions.

2.2.1 Computation

To represent the functions $\mathbf{y}(\cdot), m(\cdot), s(\cdot)$, they are *discretised* on an internal grid with coordinates

$$\{\mathbf{t}_i\}_{i=1}^{N_{grid}}, \quad t_i \in \mathbb{R}^{d_x} \tag{2.6}$$

through the use of basis functions ψ, such that the context set is encoded onto the grid via:

$$\hat{\mathbf{y}}_j = \mathbf{y}(C)(\mathbf{t}_j) = \sum_{i=1}^{N_C} \boldsymbol{\phi}\left(y_{Ci}\right) \psi_E\left(\mathbf{t}_j - \mathbf{x}_{Ci}\right) \tag{2.7}$$

Here ψ_E is the encoder basis function, which is chosen to be a squared-exponential kernel of lengthscale ℓ_E:

$$\psi_E(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell_E}\right), \tag{2.8}$$

[2]Note: ρ_θ is a map from functions $\mathbf{y}(C) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y+1}$ to functions $m : \mathbb{R}^{d_x} \to \mathbb{R}, s : \mathbb{R}^{d_x} \to \mathbb{R}$.

and

$$\boldsymbol{\phi}(y) = \left[1, y\right]^T \tag{2.9}$$

simply ensures that the model can distinguish the absence of an input from $y = 0$, and otherwise just acts as a weight.

The function ρ_θ is then applied to map the gridded input representation $\{\hat{\mathbf{y}}\}_{i=1}^{N_{grid}}$ to the gridded output representation $\{m_i\}_{i=1}^{N_{grid}}$ and $\{s_i\}_{i=1}^{N_{grid}}$.

$$\left[\hat{m}_1 \quad \dots \quad \hat{m}_{N_{grid}}\right], \left[\hat{s}_1 \quad \dots \quad \hat{s}_{N_{grid}}\right] = \rho_\theta(\mathbf{h}^{(0)}), \quad \mathbf{h}^{(0)} = \left[\hat{\mathbf{y}}_1 \quad \dots \quad \hat{\mathbf{y}}_{N_{grid}}\right]. \tag{2.10}$$

We denote any internal feature maps as $\mathbf{h}^{(i)}$. Note that $\mathbf{h}^{(i)}$ is a $(b \times a^{d_x})$-dimensional object when ρ_θ is a Convolutional Neural Network (CNN), where $a, b \in \mathbb{N}$ are specified by the model architecture. For example, if we're considering 2-dimensional space, $d_x = 2$ and feature-map resolution $a = 200$, and $b = 64$ CNN channels, each feature map consists of 200×200 values, and $\mathbf{h}^{(i)}$ consists of 64 such feature maps.

Finally, the target-point means and variances are extracted by computing a weighted sum over all gridded means/variances, with weights computed by a decoder kernel ψ_D:

$$\mu_j = m(\mathbf{x}_{Tj}, \{\hat{m}_i\}) = \sum_{i=1}^{N_{grid}} \hat{m}_i \, \psi_D \left(\mathbf{x}_{Tj} - \mathbf{t}_i\right) \tag{2.11}$$

$$\sigma_j = s(\mathbf{x}_{Tj}, \{\hat{s}_i\}) = \sum_{i=1}^{N_{grid}} \max(0, \, \hat{s}_i)\psi_D \left(\mathbf{x}_{Tj} - \mathbf{t}_i\right)). \tag{2.12}$$

Here, $\max(0, \, s_i)$ ensures positivity of the standard deviations, and ψ_D is also a squared exponential kernel (see Eq. 2.8), but with a different lengthscale ℓ_D.

The choice of hyperparameters ℓ_E, ℓ_D and N_{grid} plays an important role in the Sim2Real process and are further discussed in the experiment-specific sections 4.2 and 5.5.

For brevity, we denote the full forward pass of the model as $\mu_\theta(\mathbf{x}_{Tj}), \sigma_\theta(\mathbf{x}_{Tj})$.

2.2.2 Translation Equivariance and CNNs

By choosing ρ_θ to be a *translation equivariant* function, represented by a CNN, we ensure that predictions are also translation equivariant[3]. This inductive bias significantly constrains the space of functions ρ_θ can represent, significantly reducing overfitting and increasing sample efficiency and training speed compared to its non-convolutional counterpart. It also allows predictions to extrapolate to previously unseen areas. All of these properties are desirable for spatiotemporal modelling.

We further describe the specific ConvCNP used in this thesis, including our choice of ρ_θ in Sec. 3.2.1.

[3]Because of the internal discretisation, the represented functions are only approximately translation equivariant. With a sensible choice of grid resolution and encoder/decoder lengthscales, this approximation is good (Gordon et al., 2019).

2.2.3 Gaussian Processes vs. ConvCNPs

A commonly used spatiotemporal model, which has been used in similar applications to ConvCNPs is the Gaussian Process (GP).

In contrast to GPs, which scale cubically with the number of data points, ConvCNPs enjoy a cheap linear $\mathcal{O}(N_C + N_T)$ cost in the number of context and target points, as the former are ingested via Eq. 2.7 and the latter are simply evaluations of the mean and standard deviation functions m, s. Approximate GPs exist easing the computational requirements, but also diminishing the quality of predictions (Hensman et al., 2013).

The benefit of computationally cheap predictions of ConvCNPs comes at a (comparatively) large hunger for data. GPs excel in the sparse-data regime, because of much stronger inductive biases specified by the user: After the user has defined the GP by specifying a mean and covariance function $\mu(\cdot), k(\cdot, \cdot)$, Bayes' rule specifies how context data affect predictions. ConvCNPs do not have this built-in way of conditioning on data, they need to instead *learn it from data*.

If enough data are available, the looser inductive biases of ConvCNPs mean that the model can flexibly learn[4] what for GPs is specified by $\mu(\cdot), k(\cdot, \cdot)$, side-stepping the challenging process of specifying a "good" form for μ and k.

Sim2Real acts as a way to alleviate the hunger for real data: by leveraging the adaptability of deep-learning models, we can train strong models using modest amounts of real data as long as related simulator data are abundant.

2.2.4 Sim2Real with ConvCNPs

ConvCNPs have been used for some basic Sim2Real experiments (Gordon et al., 2019), where the ConvCNP is trained on data from a predator-prey simulator, which models how the populations of predators and their prey develop over time.

This model is then applied directly to real data (used as context data) to make predictions. The authors do not attempt to fine-tune the model's parameters on the real data and simply apply elementary data transformations to the simulator data (scaling of population sizes and time scales to align with the real data).

In this thesis, we refer to the above setup as the "sim-only" setting, which acts as a baseline for our experiments. We also investigate fine-tuning models with very small numbers of data points, which would be relevant in the predator-prey case (90 real data points).

2.3 Physics-based Weather Modelling

We now take a step back and examine the background of *weather modelling*, which is the real-world domain we apply our ConvCNPs to. To do so, we begin by summarising existing *physics-based* simulators that are the current standard of weather modelling.

[4]Technically, the ConvCNP directly learns the posterior distribution, which is different to learning the prior.

Current numerical methods for modelling global weather and climate are based on general circulation models (GCMs). GCMs first process observational measurements into an internal representation in a process called *data assimilation*, and then use physics-based differential equations to produce spatiotemporal predictions at unseen locations and/or future times (Lynch, 2008). These computations rely on an internal discretisation, the resolution of which strongly affects computational cost. Because of this discrete approximation, and because the initial conditions are not perfectly identifiable, the simulated weather values are associated with prediction errors, which compound both spatially and temporally away from observations.

Both the initial data assimilation and the simulation are very computationally expensive. The former limits how quickly new observations can be used to influence predictions and the latter limits the resolution of the predictions (Lynch, 2008). This means that while GCMs are extremely useful, they are also challenging to build (and therefore reliant on expert knowledge) and are very computationally expensive to run, both because of simulation and data assimilation steps. This, for example, inhibits their ability to quickly predict the weather over short timescales (Zhang et al., 2019; Nguyen et al., 2023). The higher the spatiotemporal resolution of the GCM, the higher the computational cost, significantly limiting the range of feasible resolutions.

2.4 Data-driven Weather Modelling

Alternatively, data-driven deep-learning models can be used to *learn* forecasting or spatial predictions directly from data, an approach that has undergone rapid improvements (with very recent examples including Lam et al. (2022); Nguyen et al. (2023); Chen et al. (2023); Andrychowicz et al. (2023a); Bi et al. (2023)), in a development called “the rise of data-driven weather forecasting” by the European Centre for Medium-Range Weather Forecasts (ECMRWF).

Once the models have undergone computationally costly training, predictions can be made on the order of seconds and on much higher spatiotemporal resolutions. Beyond the favourable computational cost, data-driven models are not limited by expert knowledge of physics-based simulators, as they can learn atmospheric phenomena directly from the data that might not be well-understood or infeasible to model numerically.

A current limitation of such models is that they often treat the best available simulation data as the ground truth on which to train (and sometimes evaluate) (Nguyen et al., 2023; Andersson et al., 2023). As we show in Sec. 5.1.1, there are still significant mismatches between the simulations and real observations, making model performance *at best* as good as the simulation. Simulation data are also spatially ($\sim$ 20km) and temporally (1 hr) coarse, limiting the training signal received by the model to this scale. A recent exception is MetNet-3 (Andrychowicz et al., 2023b), which does consider real observations during training. However, as a key limitation, MetNet-3 relies on supplemental simulator data from the physics-based HRRR simulator (Dowell et al., 2022) *at inference time*, so the timescale at which predictions can be made is limited by the data-assimilation time of HRRR.

2.4.1 ConvCNPs as Weather Models

ConvCNPs have recently shown promising results as climate and weather models (Andersson et al., 2023; Vaughan et al., 2022; Bruinsma et al., 2022). They have a number of properties that make them attractive for climate and weather modelling, which we outline via comparison to the Vision Transformer (ViT) (Dosovitskiy et al., 2020) used by other weather models such as Nguyen et al. (2023):

- ConvCNPs are inductively translation equivariant, which should make them more sample efficient than ViTs.
- ConvCNPs naturally handle off-the-grid data, which removes the need for challenging re-gridding operations for real observations. Transformers can, in theory, handle off-the-grid weather data with some modifications, but have so far focused on gridded data (e.g. Bi et al. (2022); Nguyen et al. (2023)), as the ViT architecture requires gridded inputs.
- ConvCNPs naturally quantify prediction uncertainty, which enables their use in uncertainty-sensitive problems.

We can view ConvCNPs as lying somewhere between Gaussian Processes and Vision Transformers, both in terms of inductive biases and hunger for data.

2.5 Finetuning and Foundation Models

Finally, we summarise the relationships between foundation models and fine-tuning, which is also gaining significance in weather modelling. As we investigate different fine-tuning approaches during this project, we believe our findings lay some of the groundwork for how to fine-tune weather models on real data, which will play an increasingly important role as foundation models gain significance in the field.

Over the past decade, the "pre-train and fine-tune" paradigm has been successfully used in computer vision tasks, where high-capacity models, such as ResNet (He et al., 2016) are pre-trained on large and varied databases such as ImageNet (Deng et al., 2009) and then fine-tuned to adapt to other target domains, often with much smaller datasets. The features extracted by the high-capacity models can prove very useful for the related, but slightly different, target domain, in which the model would rapidly overfit if initialised randomly.

More recently, scalable *foundation models* have reached breakthrough successes in natural language processing. Notable examples include BERT (Kenton and Toutanova, 2019), GPT (Brown et al., 2020; Bubeck et al., 2023), and PaLM (Chowdhery et al., 2022). Similar successes are found in computer vision models, such as CLIP (Radford et al., 2021) or florence (Yuan et al., 2021). In foundation models, highly advanced capabilities emerge by leveraging the structure shared across a very wide range of training tasks and the consolidation of these training tasks into unified training datasets (Bommasani et al., 2021). Because foundation models are trained on such a breadth of data, they can be easily fine-tuned to new (but related) domains: the foundation model already extracts features that are useful in the new domain.

For *small* target datasets, *adapters* have been shown to excel at fine-tuning: FiLM adapters Perez et al. (2018), for instance, represent a tiny fraction of affine parameters distributed *throughout* the model. This allows them to have an impact at every stage of the processing pipeline, without having enough capacity to rapidly overfit to the small target dataset. We explain FiLM adapters further in Sec. 3.4.1.

If the target dataset is sufficiently large, global fine-tuning of the model (i.e. tuning all parameters, including convolutional layers) becomes preferable: while adapters are less prone to overfitting, they are also less expressive than the whole model, and as the size of the target dataset grows, this tradeoff eventually falls in favour of global finetuning.

2.5.1 Foundation Models for Climate Modelling

Following breakthrough successes of foundation models in language and vision, weather and climate models such as the ClimaX foundation model (Nguyen et al., 2023) have recently followed suit. ClimaX is trained on a wide variety of different datasets and achieves comparable or superior performance to physics-based simulations in forecasting and downscaling, which is the superresolution of coarse observations by extrapolating higher frequency features between observations while interpolating the observations themselves. ClimaX has no understanding of the physics underlying weather and climate beyond the understanding *learned from data.* With foundation models such as ClimaX gaining importance in the climate and weather modelling domains, the need for an evaluation of different fine-tuning approaches grows.

ClimaX is pre-trained on a subset of the coarsely-gridded CMIP6 data collection (Eyring et al., 2016). For its spatial downscaling experiments, ClimaX transfers from this coarse grid to the finer grid of the ERA5 dataset (Hersbach et al., 2020) (see Sec. 5.1) on which it fine-tunes. The quality of predictions made by this approach is itself limited by the quality of ERA5. In this thesis, we go a step beyond, by using ERA5 for pre-training and fine-tuning to *real* data. ConvCNPs are a natural choice for this task because they naturally handle off-grid data.

2.6 Summary

ConvCNPs are spatiotemporal models that strike a good balance between data hunger and expressiveness, by employing deep learning and translation equivariance as an inductive bias. This makes them more flexible than Gaussian Processes, but less data and computation-intensive than Vision Transformers. They additionally handle off-the-grid data naturally, making them ideal candidates for spatiotemporal modelling and, crucially, fine-tuning on real data.

In weather modelling, these properties have led the ConvCNP to yield promising results, which we now want to further strengthen by overcoming the limitation of simulator-based training.

In Chapter 3, we cover available theory, our general approach, and our fine-tuning methodology used across experiments.

Chapter 3

Fine-tuning ConvCNPs

In this chapter, we explore our general approach across experiments as well as the specific model architecture. Broadly, we use large simulator-generated datasets to train a Convolutional Conditional Neural Process (ConvCNP) from random initialisation until convergence. We then use a smaller-size "real" dataset to adapt the model to the "real" domain, either by tuning parts of the model, or all of it.

3.1 Available Theory

While the work we conduct as part of this thesis is overwhelmingly empirical in nature, we try to formalise our procedure mathematically in this section, based on the limited available theory.

In the following sketch proof, we show that by minimising the model's negative conditional log-likelihood during training, we minimise the KL Divergence between the model's predictive and the true predictive distribution, as long as the number of training samples is large.

We would like to model the *true posterior predictive* distribution over target observations $\mathbf{y}_T$, given target locations X_T and a set of context data $C = \{(\mathbf{x}_{Ci}, y_{Ci})\}_{i=1}^{N_C}$:

$$p(\mathbf{y}_T|X_T, C). \tag{3.1}$$

Because we do not have access to the *true* posterior predictive, we model an *approximate posterior predictive* distribution as a ConvCNP $q_\theta(\mathbf{y}_T|X_T, C)$, parameterised by θ (for more detail see Sec. 2.2).

The quality of the approximate posterior predictive distribution is measured by the expected KL divergence between the true and approximate posterior predictives, under the true distribution of observations $p(X_T, \mathbf{y}_T, C)$. In other words, the goal is to find a set of parameters θ^* that minimises this KL divergence:

$$\theta^* = \arg\min_\theta \mathbb{E}_{p(X_T,C)} \left[\mathrm{KL}\left[p(\mathbf{y}_T|X_T,C) \mid\mid q_\theta(\mathbf{y}_T|X_T,C)\right]\right] \tag{3.2}$$

$$= \arg\min_\theta \; -\mathbb{E}_{p(X_T,C)} \left[p(\mathbf{y}_T|X_T,C) \log \frac{q_\theta(\mathbf{y}_T|X_T,C)}{p(\mathbf{y}_T|X_T,C)}\right] \tag{3.3}$$

$$= \arg\min_\theta \; -\mathbb{E}_{p(X_T,C)} \left[p(\mathbf{y}_T|X_T,C) \log q_\theta(\mathbf{y}_T|X_T,C)\right] \tag{3.4}$$

$$= \arg\min_\theta \; -\int p(\mathbf{y}_T|X_T,\lambda)p(\lambda|C)p(C) \log q_\theta(\mathbf{y}_T|X_T,C) d\lambda\, dC\, dX_T\, d\mathbf{y}_T \tag{3.5}$$

$$= \arg\min_\theta \mathbb{E}_{p(X_T,\mathbf{y}_T,C)} \left[-\log q_\theta(\mathbf{y}_T|X_T,C)\right] \tag{3.6}$$

$$= \arg\min_\theta \mathcal{L}. \tag{3.7}$$

Here, λ represents the latent variables of the task, such as the state of the atmosphere for weather modelling, and is informed by the context set C. The measurements at the target locations $\mathbf{y}_T$ are *independent* of the context set C given the latent variables λ. Throughout our experiments, we denote this loss as $\mathcal{L}$.

To approximate this expectation value over the negative log-likelihood, we use our available training data as samples from the true joint distribution $p(X_T, \mathbf{y}_T, C)$:

$$\mathcal{L} = \mathbb{E}_{p(X_T,\mathbf{y}_T,C)} \left[-\log q_\theta(\mathbf{y}_T|X_T,C)\right] \tag{3.8}$$

$$\approx -\frac{1}{N_{data}} \sum_{n=1}^{N_{data}} \log q_\theta(\mathbf{y}_T^{(n)}|X_T^{(n)}, C^{(n)}), \quad X_T^{(n)}, \mathbf{y}_T^{(n)}, C^{(n)} \sim p(X_T, \mathbf{y}_T, C) \tag{3.9}$$

$$= -\frac{1}{N_{data}} \sum_{n=1}^{N_{data}} \log \prod_{i=1}^{N_T^{(n)}} \mathcal{N}(y_i^{(n)}|\mu_\theta(\mathbf{x}_i^{(n)}|C^{(n)}), \sigma_\theta^2(\mathbf{x}_i^{(n)}|C^{(n)})) \tag{3.10}$$

$$= -\frac{1}{N_{data}} \sum_{n=1}^{N_{data}} \sum_{i=1}^{N_T^{(n)}} \log \mathcal{N}(y_i^{(n)}|\mu_\theta(\mathbf{x}_i^{(n)}|C^{(n)}), \sigma_\theta^2(\mathbf{x}_i^{(n)}|C^{(n)})). \tag{3.11}$$

In this notation, $n = 1 \ldots N_{data}$ indexes the available training tasks, consisting of context set $C^{(n)}$, as well as $N_T^{(n)}$ target locations $\mathbf{x}_i^{(n)}$ and target observations $y_i^{(n)}$. We can factorise the distribution over all target points because ConvCNPs only model uncorrelated target observations. This loss is optimised via standard mini-batch gradient descent as described in Sec. 3.3.

In our context, this means that the pre-trained model is well-equipped to capture the predictions implied by the simulator (from which the pre-training data originates). In the fine-tuning phase, however, the training set is smaller and therefore likely a poor approximation of the true distribution. In this regime, regularisation is needed to prevent overfitting.

3.1.1 Sim2Real Theory

In the Sim2Real setting, we have two “true” joint distributions, the *simulator* joint distribution $p^{sim}(X_T, \mathbf{y_T}, C)$, and the *real* joint distribution $p^{real}(X_T, \mathbf{y_T}, C)$.

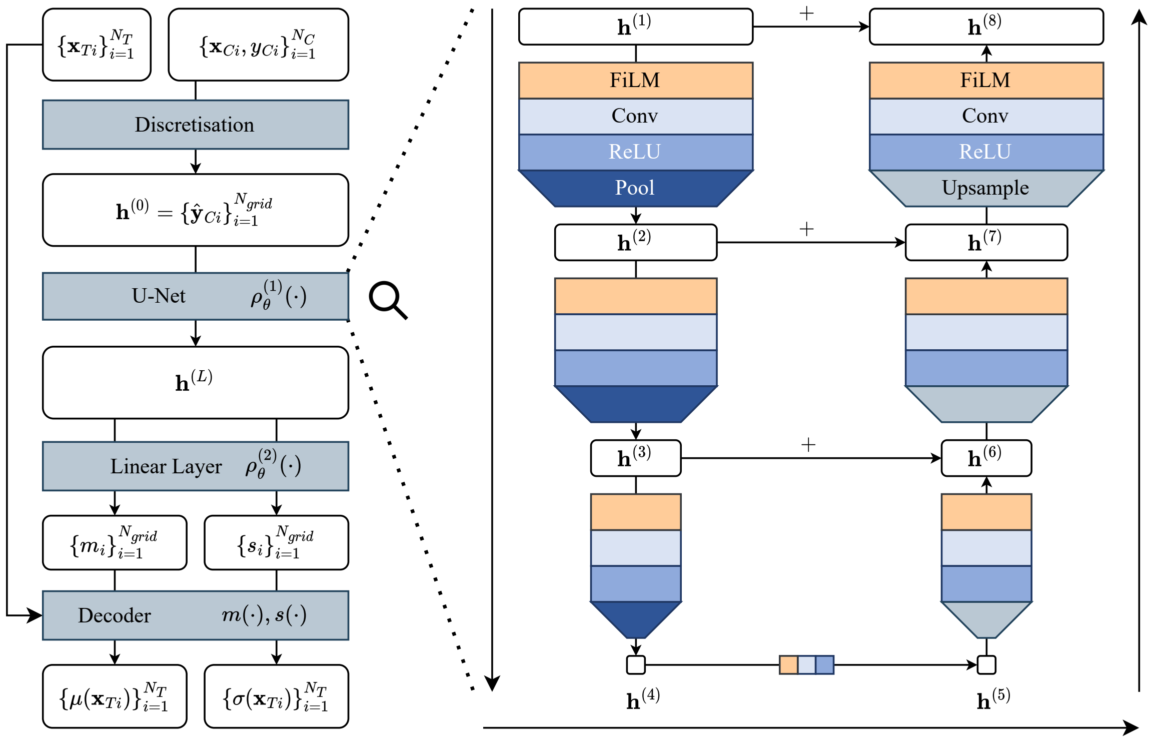

$\{\mathbf{x}_{Ti}\}_{i=1}^{N_T}$
$\{\mathbf{x}_{Ci}, y_{Ci}\}_{i=1}^{N_C}$
Discretisation
$\mathbf{h}^{(0)} = \{\hat{\mathbf{y}}_{Ci}\}_{i=1}^{N_{grid}}$
U-Net $\rho_\theta^{(1)}(\cdot)$
$\mathbf{h}^{(L)}$
Linear Layer $\rho_\theta^{(2)}(\cdot)$
$\{m_i\}_{i=1}^{N_{grid}}$
$\{s_i\}_{i=1}^{N_{grid}}$
Decoder $m(\cdot), s(\cdot)$
$\{\mu(\mathbf{x}_{Ti})\}_{i=1}^{N_T}$
$\{\sigma(\mathbf{x}_{Ti})\}_{i=1}^{N_T}$
$\mathbf{h}^{(1)}$
$\mathbf{h}^{(8)}$
FiLM
Conv
ReLU
Pool
Upsample
$\mathbf{h}^{(2)}$
$\mathbf{h}^{(7)}$
$\mathbf{h}^{(3)}$
$\mathbf{h}^{(6)}$
$\mathbf{h}^{(4)}$
$\mathbf{h}^{(5)}$
+

Figure 3.1: The ConvCNP architecture, using a U-Net (Ronneberger et al., 2015) (and linear layer) as ρ_θ. Coloured boxes with sharp corners are the processing steps and rounded white boxes are the different states along the processing pipeline.

The goal is to find a set of parameters θ^{real} that minimises the KL divergence between q_θ and p^{real}, but potentially given only a small number of tasks $\tau^{real} \sim p^{real}$. Given that the simulator tries to approximate the real process, we assume there is a large amount of shared structure between p^{real} and p^{sim}.

In the case of global fine-tuning (tuning all parameters $\in \theta$), we can then write

$$\theta^{real} = \theta^{sim} + \delta\theta, \tag{3.12}$$

and only need to learn the small[1] $\delta\theta$ from the limited real data.

In the case of FiLM adaptation (see Sec. 3.4.1), we can instead describe the process as tuning one set of parameters θ_0, which is shared between the "sim" and the "real" models, and adjusting a separate, smaller, set of parameters θ' for encoding domain-specific differences in the model. Here θ_0 might be the parameters of the convolutional filters and final linear layer (light blue and grey in Fig. 3.2.1), and θ' the parameters of the FiLM layers (orange in Fig. 3.2.1).

3.2 Model

3.2.1 Architecture

We describe the ConvCNP used throughout this thesis in Sec. 2.2. The full architecture of the model is shown in Fig. 3.1. To completely specify the model, it only remains to define the implementation of ρ_θ and the hyperparameters of the model.

The specific convolutional architecture we choose for ρ_θ is the U-Net (Ronneberger et al., 2015), which is depicted in Fig. 3.1 (right). The U-Net is a common backbone for ConvCNPs (Gordon et al., 2019; Andersson et al., 2023; Vaughan et al., 2022), because it first exponentially reduces the size of the feature maps on the downward-leg by consecutively halving feature map sizes with pooling layers, and then exponentially increases the size of the feature maps back to the original size by consecutively using up-sampling layers. This allows the model to consider a wide receptive field using small filters, with higher-up layers capturing short-range dependencies and lower-down layers capturing long-range dependencies. To avoid information loss, residual connections transfer information at every level of the downward leg to the corresponding level of the upward leg of the "U".

We follow this U-Net with a single linear layer for final outputs of gridded means and variances m_i, s_i.

Hyperparameters, including the number of layers and layer capacities, are chosen on a per-experiment basis and described in the experiment-specific sections 4.2 and 5.5.

3.3 Optimisation

Across experiments, we share the following optimisation strategy:

- We use the Adam optimiser (Kingma and Ba, 2014) to compute weight updates.
- On the abundant *simulator* data, we pre-train the model until convergence, starting at a learning-rate α and reducing it by a factor of 3 when the validation loss has not improved for a specified number of epochs (called the patience).
- Starting from the pre-trained parameters, we fine-tune the model on the limited *real* data, holding out a fraction for validation. This step might benefit from different hyperparameters to the pre-training step, so we re-tune hyperparameters for each experiment[2].

3.4 Finetuning Approaches

Our fine-tuning approaches can be categorised as one of the following:

[1]Small when compared to random initialisation, i.e. random θ^{sim}.

[2]We have found this to be a particularly time-consuming process.

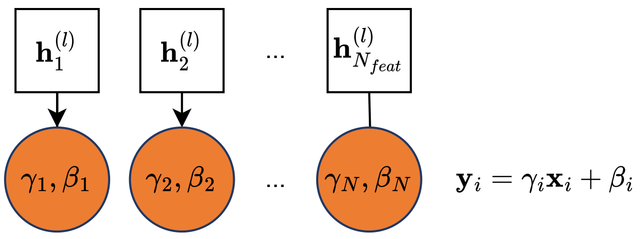


Figure 3.2: FiLM adapters (Perez et al., 2018) scale and shift each feature map $\mathbf{h}_i^{(l)}$ without combination with other feature maps.

1. Global fine-tuning (fine-tuning everything),
2. Freezing cohesive sections of the model, such as the long-range U-Net layers, and tuning the rest,
3. Tuning only the FiLM layers throughout the model (see Sec. 3.4.1).

3.4.1 FiLM

When finetuning models, large sections of the model are commonly frozen (such as the feature-extractor in visual models) to restrict the number of parameters trained using the potentially small finetuning dataset. A possible issue with this approach is that the large frozen sections might not be extracting optimal features for the downstream task.

FiLM adapters (Perez et al., 2018), visualised in Fig. 3.2, are a light-weight alternative method of adapting models and are commonly applied *throughout* the model. FiLM adapters operate on a single feature map $\mathbf{h}_i^{(l)}$ by transforming it affinely, i.e. scaling it by a factor γ_i and shifting it by a factor β_i:

$$\tilde{\mathbf{h}}_i^{(l)} = \gamma_i^{(l)} \times \mathbf{h}_i^{(l)} + \beta_i^{(l)}. \tag{3.13}$$

Here, l indexes the layer of the deep-learning model and i indexes the feature maps generated by that layer. Because a FiLM layer contains only $2N_{feat}$ trainable parameters, where N_{feat} is the number of feature maps in the layer, they have low capacity (and can therefore be adapted using few training samples) but can affect features throughout the model (as opposed to freezing sections of the model to reduce capacity). FiLM has shown strong results in data-efficient adaptation (Perez et al., 2018; Gupta and Brandstetter, 2022). We apply FiLM adapters after every convolutional layer as shown in Fig. 3.1 (orange).

During pre-training, we fix $\beta_i^{(l)} = 0, \gamma_i^{(l)} = 1$ instead of training them. During finetuning, we then freeze all other model parameters and train only the FiLM parameters. This allows for a slightly larger capacity of the FiLM layers (as they are not already part of the feature generation).

3.4.2 Other Fine-Tuning Approaches

We also experiment with freezing sections of the model, treating lower-level layers as feature extractors and simply fine-tuning higher layers. This is a popular way of fine-tuning foundation models in computer vision that reduces the number of trainable parameters and computational cost of backpropagation and has been investigated for fine-tuning the ClimaX climate foundation model (Nguyen et al., 2023), but we find no success with the method ourselves (in either experiment).

We also combine different approaches, such as training just the linear layer (see Fig. 3.1) and FiLM layers to increase the capacity of fine-tuned parameters, but again find no success across our experiments. We, therefore, do not dedicate detailed sections to these alternative methods in the experiment-specific sections 4 and 5.

Chapter 4

Evaluation: Gaussian Progress Regression

In this synthetic experiment, we perform “Sim2Real” by generating both the “simulated” and the “real” data. In this simple setup, we control both ends of the Sim2Real process, allowing us to accurately define baselines and to roughly control the size and qualitative nature of the Sim2Real gaps. In this simplified setup, we can iterate quickly and gather a wide range of results, acting as a rough proxy for real Sim2Real applications.

4.1 GP Data

Specifically, we consider a Gaussian Process (GP) $\mathcal{GP}_\ell$ (Williams and Rasmussen, 2006). $\mathcal{GP}_\ell$ is specified by a 0 mean function and squared-exponential covariance function

$$k_\ell(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell}\right), \tag{4.1}$$

of lengthscale ℓ. We then train a Convolutional Conditional Neural Process (ConvCNP) (see Sec 2.2) to emulate the GP (see Fig. 4.1) by training it on context and target data C, T generated by the GP.

Each task is drawn from the GP independently from other tasks:

$$f(x) \sim \mathcal{GP}(0, k_\ell). \tag{4.2}$$

Specifically, we uniformly draw a number of context points N_C, and choose the number of target points N_T:

$$N_C \sim \mathcal{U}_{\text{disc}}(\lfloor 1/\ell \rfloor, \lfloor 10/\ell \rfloor), \quad N_T = \lfloor 15/\ell \rfloor, \tag{4.3}$$

where the $1/\ell$ dependence ensures that the training set covers short lengthscales (and not just uncorrelated samples) often enough to not impair training.

Then, we draw the (1D) x-coordinates for the context and training sets uniformly[1]:

$$x_{Ci} \sim \mathcal{U}(-2, 2), \quad x_{Tj} \sim \mathcal{U}(-2, 2), \tag{4.4}$$

[1]The -2 to 2 range is arbitrary and only matters in relation to other lengthscales.

along with noisy observations at these coordinates:

$$y_{Ci} = f(x_{Ci}) + \epsilon_i, \; y_{Tj} = f(x_{Tj}) + \epsilon_j, \quad \epsilon_i, \epsilon_j \sim \mathcal{N}(0, \sigma_0), \tag{4.5}$$

for all $i \in 1 \dots N_C$ and $j \in 1 \dots N_T$, giving us a task:

$$C = \{(x_{Ci}, y_{Ci})\}_{i=1}^{N_C}, \; T = \{(x_{Tj}, y_{Tj})\}_{j=1}^{N_T}. \tag{4.6}$$

A dataset $\mathcal{D}$ consists of multiple such tasks

$$\mathcal{D}_{\ell,\sigma_0,N_{tasks}} = \{T, C\}_{n=1}^{N_{tasks}}, \tag{4.7}$$

and is solely specified by the GP lengthscale ℓ, the level of observation noise σ_0, and the number of tasks N_{tasks}.

All of our experiments in this section consider a model pre-trained using one dataset $\mathcal{D}^{sim}$ with $\ell^{sim}, \sigma_0^{sim}$ with infinite[2] N_{tasks}, and then finetuned using a *different* dataset $\mathcal{D}^{real}$ with ℓ^{real} and σ_0^{real} of *limited* N_{tasks}^{real}.

4.2 Experimental Details

Model Hyperparameters We use the model architecture described in Sec. 2.2 with 5 layers (down and up) in the U-Net, of 64 channels each. We choose a resolution of 64 Points Per Unit (PPU) for the internal gridded representation and a corresponding encoder lengthscale $\ell_E = \ell_D = 1/64$, which allows us to comfortably resolve the lowest-lengthscale features of our experiments (the shortest $\ell^{real} = 0.05$ needs resolutions of $\approx 1/0.05 = 20$PPU or higher to be resolved on the internal grid). We found training ℓ_E and ℓ_D to not be beneficial. In total, our model has 300,000 parameters, of which 1700 (0.6%) are FiLM parameters. We also experiment using higher and lower capacity models but found that this capacity yields close to optimal performance at a relatively small size.

We compute the convolutional kernel size s_{cnn} corresponding to a receptive field of 1.2 units to capture the largest lengthscales during our experiments $\ell^{real} = 1.0$ given the 5 U-Net layers as:

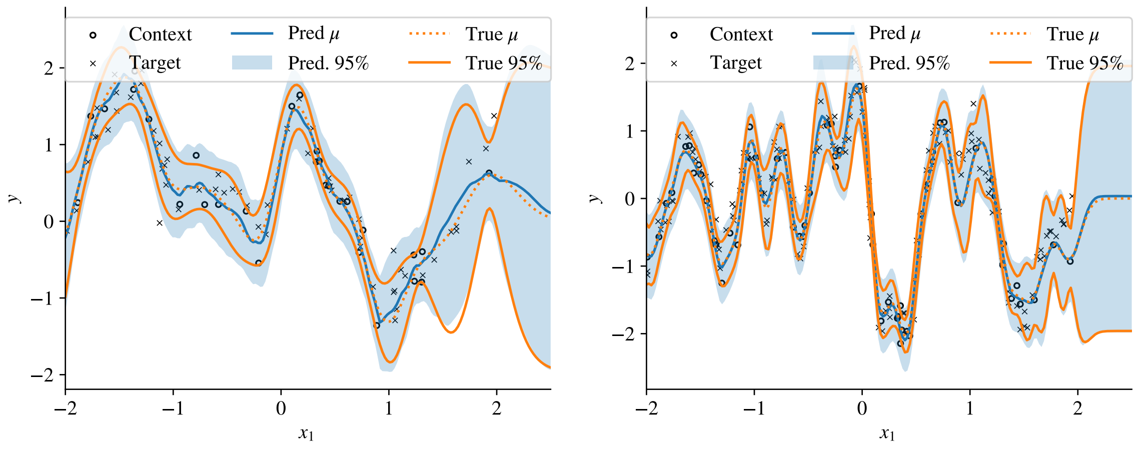
$$s_{cnn} = \lceil \frac{1.2 \times 64}{2^{5-1}} \rceil = 5. \tag{4.8}$$

To avoid checkerboard artefacts encountered during training, we follow Odena et al. (2016) and use bilinear resize convolution layers, which resolved the issue.

Optimisation We use the Adam optimiser (Kingma and Ba, 2014), with a learning rate of 1×10^{-4} and a batch size of 16 during pre-training. These hyperparameters were determined via (exponential) grid-search using roughly factors of 3 for the learning-rate[3] and 2 for the batch-size between $[1 \times 10^{-5} \; 1 \times 10^{-2}]$ and [4, 64] respectively, though given that we have infinite available training data in the "simulator"-phase we

[2]In practice, we generate as many new tasks as required for convergence on a held-out validation set.

[3]I.e. we covered learning rates of 1×10^{-5}, 3×10^{-5}, 1×10^{-4}, ...



(a) $\ell = 0.25$ (b) $\ell = 0.1$

Figure 4.1: Two samples from GPs with different lengthscales (orange), and predictions made by trained ConvCNPs (blue). The small mismatches are due to the assumption of marginal target variances implicit in ConvCNPs (see Sec. 2.2)

did not find significant differences in final performance, only in convergence speed. We also annealed the learning rate by a factor of 3 if the validation loss stalled for more than 5 epochs, which allows for complete convergence to a local optimum. We train until convergence using 1024 tasks per "epoch". During finetuning, we hold out 25% of the available tasks for validation and early stopping to prevent overfitting.

We took care to tune the learning rate α for the fine-tuning experiments, as the differences between different fine-tuning methods are very small: In general the smaller the Sim2Real gap and the smaller N^{real}_{tasks}, the smaller α needs to be to "catch" the optimal early-stopping time[4]. For FiLM adaptation, α can be significantly larger ($\sim$ by a factor of 50), because the fewer parameters overfit much less quickly. To avoid clutter, we put the experiment-specific learning rates in Appendix A.

4.3 Shrinking Lengthscales

In the first experiment, we keep noise fixed at $\sigma_0^{real} = \sigma_0^{sim} = 0.05$ and consider the transfer from $\ell^{sim} = 0.25$ to *shorter* lengthscale GPs, with ℓ^{real} of either 0.2, 0.1 or 0.05. This is analogous to the target domain of weather modelling (see Sec. 5), where some real measurement stations are closely separated ($\ell^{real} \sim 4$km) and can therefore capture shorter lengthscale weather phenomena than the more coarsely gridded ERA5 Reanalysis (ERA5) simulator data with $\ell^{sim} \sim 20$km grid-spacing[5].

[4]Often, the optimal early stopping time was shortly after the first full epoch, and if the learning rate is too large, by the second epoch the optimum had passed.

[5]The analogy is not perfect, as long-lengthscale weather phenomena are still present in both real and simulator data, which is not the case for these GPs, but a part of the problem is captured nonetheless.

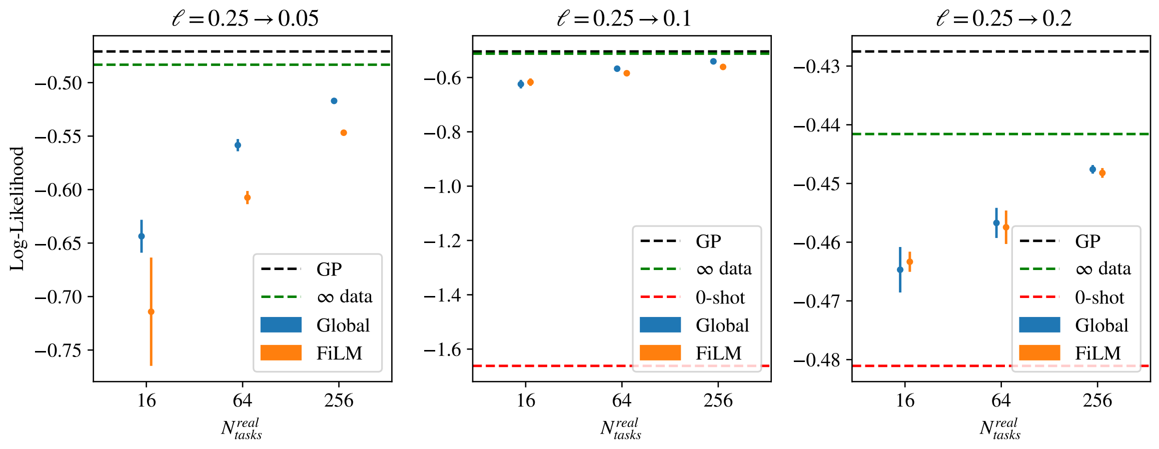


Figure 4.2: Test-set log-likelihoods achieved via fine-tuning on limited numbers of "real" tasks (x-axis). Global fine-tuning outperforms FiLM in the shrinking lengthscale experiments, particularly if the difference in lengthscales is large. FiLM performs slightly worse the more fine-tuning data are available. Error bars represent 95% confidence intervals and are computed by starting from the same pre-trained model and using different fine-tuning datasets. The 0-shot baseline in the left-most plot is ≈ -4.1 and is hidden to not distort the y-scale.

As shown in Fig. 4.2, both FiLM and global fine-tuning require only a very small number of tasks for effective adaptation to shorter lengthscales, when compared to the (poor) 0-shot baseline. In the more extreme $\ell = 0.25 \to 0.05$ transfer, global fine-tuning significantly outperforms FiLM. We hypothesise that the convolutional filters learned on the pre-training task extract features that are tuned to the particular $\ell^{sim} = 0.25$ – for less extreme changes in ℓ, FiLM adaptation is able to scale features to achieve similar performance to global fine-tuning, for much smaller lengthscales, the features extracted by the convolutional filters become less useful and fine-tuning the filters themselves becomes important.

Overall, this effect is smaller for smaller values of N^{real}_{tasks}. In these sparse-data regimes, the much lower capacity of FiLM adaptation gives the model a lesser opportunity to overfit. For $\ell^{real} \in (0.1, 0.2)$, and $N^{real}_{tasks} = 16$, this leads to FiLM slightly outperforming global finetuning.

4.4 Growing Lengthscales

If our hypothesis that FiLM layers cannot rectify low-resolution convolutional filters holds, we should see an improved FiLM performance in the inverse problem of growing lengthscales. This setting is less analogous to the real weather modelling experiment (Sec. 5), but is useful to include for generality and a more domain-agnostic approach to fine-tuning.

We therefore now consider $\ell^{sim} = 0.2$ and $\ell^{real} \in [0.25, 0.5, 1.0]$. As hypothesised, FiLM performs slightly better in this setting (see Fig. 4.3), particularly in the sparse-

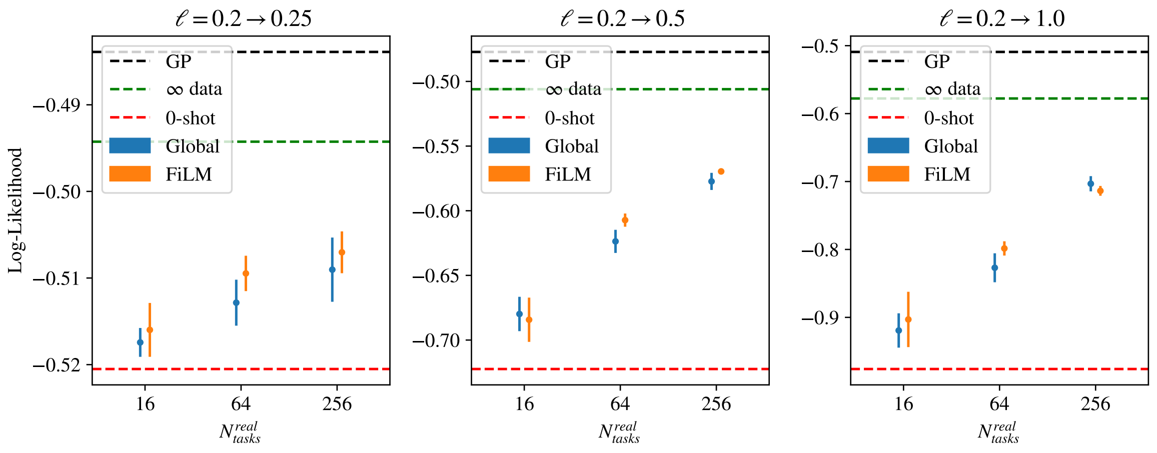


Figure 4.3: For growing lengthscales, FiLM outperforms global fine-tuning. Especially for sparse data settings and small Sim2Real gaps.

data regime.

4.5 Multi-lengthscale Tuning

We also attempt to pre-train our ConvCNP on a range of lengthscales, by sampling $\ell^{sim} \sim \mathcal{U}(0.25, 0.5)$ in our task generation. This leads to one model that is as capable at fitting to a particular lengthscale within [0.25, 0.5] as any model trained on that lengthscale (given that the model capacity is sufficient).

We hoped that by training the model to function well across a range of lengthscales, it would be able to transfer better to outside this range. However, we found no benefit to performance, whether in the 0-shot $N_{task}^{real} = 0$ setting or after fine-tuning. In fact, we found this model to be more data-hungry to fine-tune than its constant $\ell^{sim} = 0.25$ counterpart, even to the close-by lengthscale $\ell^{real} = 0.2$.

In future work, it could be interesting to explore drawing ℓ^{sim} from a different distribution, e.g. $\ell^{sim} \sim \mathcal{N}(\mu_\ell, \sigma_\ell^2)$, and exploring how the model then transfers to lengthscales $\ell^{real} \ll \mu_\ell/\sigma_\ell$, because then the pre-training assigns a non-zero probability to such lengthscales, as opposed to the uniform distribution in this experiment, though our uniform results lead us to believe that the model will not be able to generalise past lengthscales it has encountered frequently enough (and recently enough) during pre-training.

4.6 Noise Change

Finally, we keep the lengthscales fixed at $\ell^{sim} = \ell^{real} = 0.25$, and instead change the level of *noise* from $\sigma_0^{sim} = 0.05$ both up to $\sigma_0^{real} \in [0.1, 0.2]$ and down to $\sigma_0^{real} \in [0.0125, 0.025]$.

This is analogous to our weather-based Sim2Real experiments, where we would expect our simulated data, ERA5 (Hersbach et al., 2020), to be associated with lower

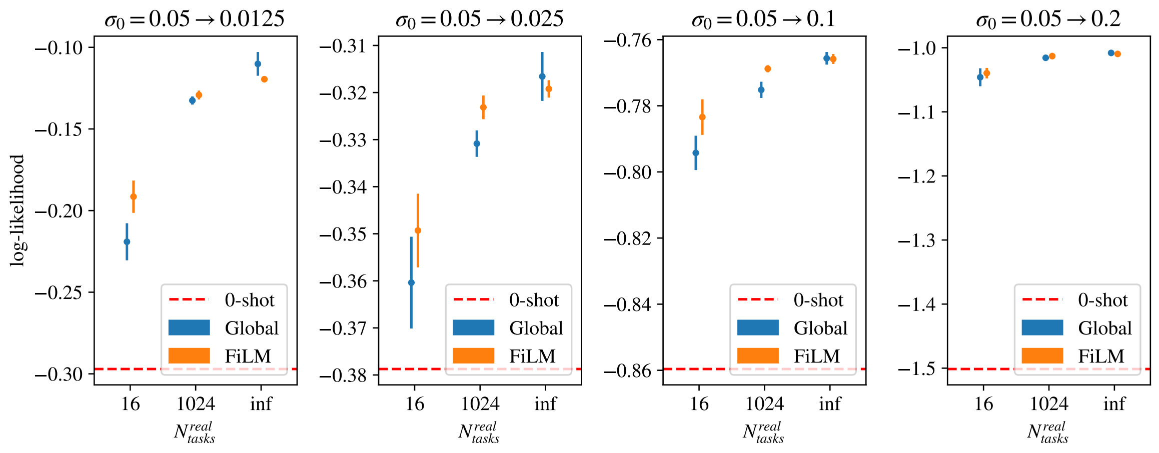


Figure 4.4: When adapting to different noise levels, FiLM adapters beat global fine-tuning decisively. Even in the infinite data limit, FiLM is only very slightly weaker.

noise[6] than real measurements because

- The simulation runs on a discrete spatiotemporal grid and therefore cannot model weather phenomena beyond its resolution.
- The data assimilation process can ingest multiple data points of observational data within one grid cell, smoothing out local measurement noise.

For generality, we both simulate an increase and a reduction in noise.

In Fig. 4.4, we show that in this regime FiLM outperforms global fine-tuning across different values of σ_0^{real}, even in the larger $N_{tasks}^{real} = 1024$ experiments. In the limit $N_{tasks}^{real} \to \infty$, global fine-tuning still outperforms (as it should), but not by a large margin.

These results align with our previous findings, that FiLM is a more sample-efficient fine-tuning method *unless* the frozen convolutional filters extract features of an insufficient resolution.

4.7 Summary

These synthetic GP regression experiments allowed us to evaluate different ways of fine-tuning ConvCNPs and expose some tradeoffs between the methods, as well as their performances in different data-availability regimes.

Overall, 0-shot performance is poor and fine-tuning yields dramatic improvements, even if only a small number of fine-tuning samples are available. FiLM adaptation generally outperforms global fine-tuning in sparse-data regimes, *unless* the convolutional filters extract inappropriate features, e.g. because of shrinking lengthscales.

[6] By noise we do not mean (negligible) inaccuracies in the temperature measurement, but instead the aleatoric uncertainty due local weather phenomena (e.g. a cloud flying overhead at the time of measurement or a cold breeze passing by) that might lead to temperature changes on the order of seconds and metres, which are infeasible to model.

We now focus on the much more difficult *temperature downscaling* experiment, which has real-world applications, to demonstrate the usefulness of Sim2Real in real settings.

Chapter 5

Evaluation: Temperature Downscaling

In this experiment, we train a ConvCNP to perform *spatial temperature downscaling*. Here, the goal is to predict the air temperature at 2 metres above ground y_{Ti} (from here on referred to simply as the temperature) at any target locations $\mathbf{x}_{Ti}$ given the temperature y_{Cj} at some context locations $\mathbf{x}_{Cj}$. Beyond training the ConvCNP on the ERA5 simulator-dataset (see Sec. 5.1), which has been performed before (Andersson et al., 2023; Vaughan et al., 2022), we fine-tune it on real stations to perform Sim2Real transfer, overcoming (some of) the approximations made by the simulator. A sample of simulator data and real data, of the same date and time, is shown in Fig. 5.1.

This is a challenging setting for a few reasons:

- The dataset is large: we use $\sim 100,000$ different training times during pre-training, each consisting of up to ~ 600 context and target points that we can each yield a very large number of "distinct" tasks (based on which points are used as context and which as target).
- During fine-tuning, we must be very careful with how to split our available data to avoid pitfalls. This is explained in detail in Sec. 5.4.
- Finally, as we move from an on-the-grid simulator to off-the-grid real data, the configuration of the ConvCNP becomes significantly more complicated than if we stayed in one regime. This is explained in Sec. 5.5.

Models often treat simulators as their ground truth because observational data are limited, especially in remote areas, and because existing observational data are owned, handled, and distributed by different organisations in different formats, at different spatiotemporal resolutions and sparsities (e.g. sparse weather stations vs. dense satellite-based observations) and measured under different protocols. It can therefore be difficult to train a model solely on observational data. We, therefore, focus exclusively on Germany for this experiment. Germany has a very high density of weather stations with publically available and reliable measurements at high temporal frequencies. This allows us to run experiments using a large number of stations and using a lot of temporal data points and gives us the freedom to select a large test set. Still, we also run all experiments in artificially scarce regimes for generality.

Weather modelling is a difficult task, in which traditional spatiotemporal models, such as GPs struggle (Andersson et al., 2023). To achieve good results, we require large

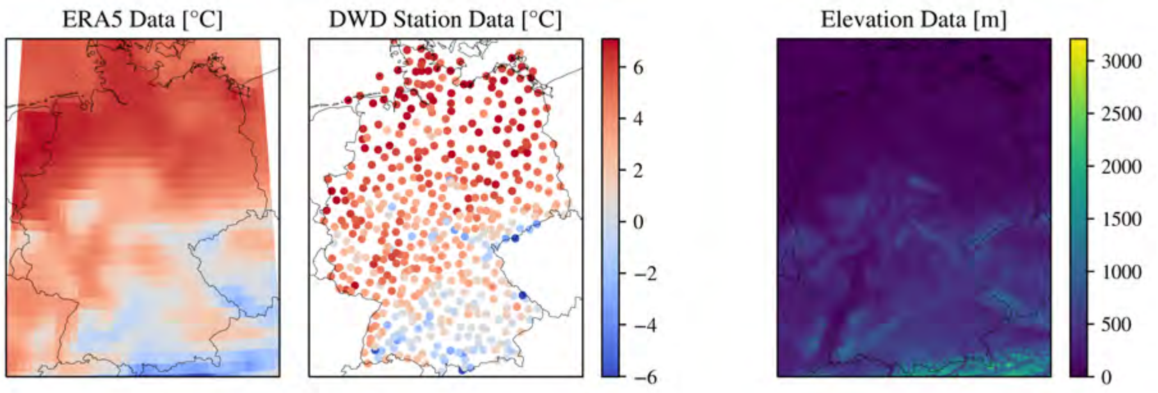


Figure 5.1: Left: A sample of the gridded ERA5. Centre: a sample of the data collected by German weather stations. Right: High-res elevation data injected to aid predictions. During Sim2Real, we transfer from the left to the central data source.

models that are slow to train (and therefore slow to iterate on), and that are associated with a large number of design choices, from model architecture and hyperparameters to the way in which we split available data into training tasks. Under the given time constraints, we choose to execute this experiment thoroughly, instead of attempting to run a third experiment. In future work, a third experiment would still be desirable to validate our results and to make them less domain-specific.

In the following sections, we outline the simulator data we use and their differences to real data (Sec. 5.1), the real temperature and auxiliary data, and how we split them into training tasks (Secs. 5.2-5.4). After discussing experimental details and baselines (Secs. 5.5, 5.6), we present quantitative results in different data-availability regimes in Sec. 5.7 and associated qualitative evaluations in Secs. 5.8, 5.9. In the subsequent Secs. 5.10, 5.15, we discuss some of the observed limitations and proposed solutions before finally investigating how Sim2Real affects a downstream task of placing new weather stations in Sec. 5.12.

5.1 Simulator Data: ERA5 Reanalysis

For pre-training, we use the gridded, physics-based ERA5 data. ERA5 is a *reanalysis* dataset, which means it simulates weather variables *in the past*. To do so, it assimilates not only past observations to make predictions at unseen times and locations, but also "present" and "future" ones, allowing it to be more accurate than *predictive* physics-based models. ERA5 is commonly used for the training of data-driven models (Andersson et al., 2023; Vaughan et al., 2022; Nguyen et al., 2023).

ERA5 is available at hourly intervals and a spatial resolution of $\sim$ 20km (amounting to approximately 600 grid points within Germany). A sample of gridded ERA5-data is shown on the left of Fig. 5.1. For training, we use hourly data between 2012-01-01 and 2020-12-31. For validation, we consider the year 2021 and for testing the year 2022. This is a minor limitation as it does not negate distribution shifts on macroscopic timescales between the training and val/test data, but it does allow us to easily and

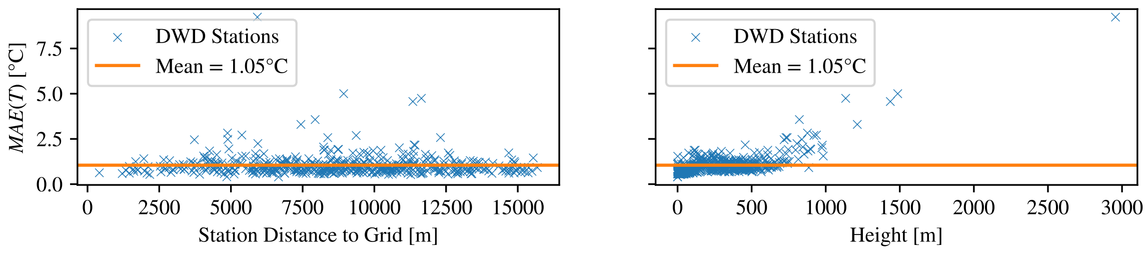


Figure 5.2: Mean absolute error (MAE) between real station measurements and closest ERA5 gridpoint, grouped by station. **Left**: MAE and distance to ERA5 gridpoints are not strongly correlated, indicating that ERA5 data are not strongly constrained by station measurements. **Right**: Higher errors at some stations are well explained by station elevation being different to coarse ERA5-internal gridded elevation.

cleanly separate the pre-training dates from the fine-tuning dates to avoid leakage (see Sec. 5.1.1). We also do not believe distribution shift to be a significant issue, as the very large amount of training data leads to minimal overfitting, and training loss qualitatively matches validation loss well during training.

Because of the real data assimilated into ERA5, we worried about the leakage of station data into the model that might lead to the model performing disproportionately well at the stations, diminishing the validity of our evaluations. In the following Sec. 5.1.1, we show that this is unlikely to have an impact. We take the further precaution of not evaluating the model on any real stations at times after 2021-01-01, i.e. when the simulator pre-training ends, to prevent temporal leakage.

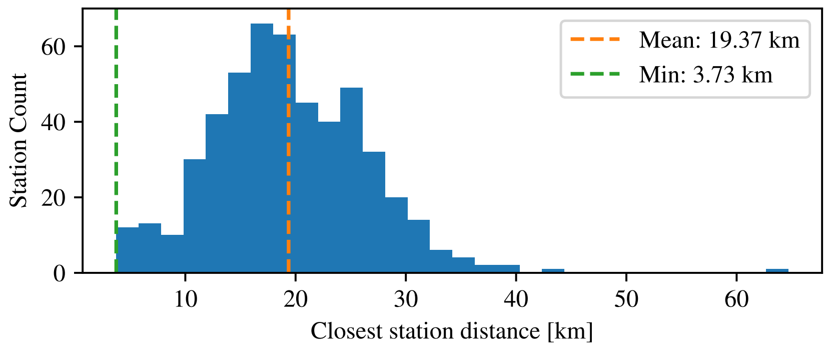
5.1.1 Differences to Real Data

We ultimately want to evaluate our model on station data, but because ERA5 is a reanalysis model, station observations could have entered the simulator data through the reanalysis process. If constraints were strong, and thus the simulator very well-aligned with station data, we would be at risk of real evaluation data leaking into our simulator's pre-training set. The model trained on this pre-training set could then spatially and temporally overfit to the station data (via the ERA5 data), leading to a biased evaluation.

As ERA5 makes use of a wide range of data sources for reanalysis (Hersbach et al., 2020), predominantly satellite and radar data, we do not expect this to be a significant issue, but investigate it nonetheless.

We prevent temporal leakage by selecting non-overlapping time ranges for our pre-training and fine-tuning data. To ensure the model cannot overfit spatially through ERA5, we investigate the discrepancies between ERA5 and real data at the same times: As shown in Fig. 5.2, stations closer to the ERA5 grid points are associated with similar discrepancies from ERA5 as stations away from the grid points, indicating that ERA5 is not strongly constrained by said station observations. This means spatial overfitting to stations is, if at all present, negligible.

Fig. 5.2 provides another insight into ERA5: the Mean Absolute Error (MAE)



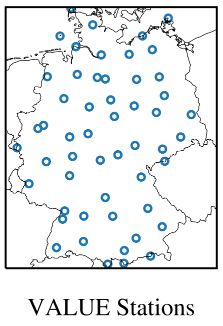


Figure 5.3: Left: Stations are mostly within 30 kilometres of other stations. The closest inter-station separation is 3.73km. Right: The VALUE stations used for testing.

of the ERA5 data is 1.05°C, with some stations showing much larger discrepancies because the limited spatial resolution of ERA5 can not account for elevation well, which is particularly problematic in the alps. This means any model trained on this data can, at best, achieve mean absolute errors of $\sim$ 1.05°C. In reality, the model cannot perfectly imitate the simulator either, leading to compounding errors, further highlighting the utility of Sim2Real.

5.2 Real Data: German Weather Stations

We select Germany for our experiments because it has a large number of weather stations, which allows us to experiment in high data-availabity regimes. These weather data are collected and made publically available by the German Weather Service (DWD) climate data centre (Wetterdienst, 2023). Another major benefit is that *snapshot data*, i.e. temperature measurements recorded at a specific latitude, longitude and *time* are consistently available. In more remote regions, such as Antarctica, data are less consistently available and similar work has therefore focused on *daily-averaged* data e.g. (Andersson et al., 2023). By focusing on snapshot data throughout this thesis, we focus on a setting that is

- More difficult to model than daily averages, because daily averaging smooths out short timescale weather phenomena[1], and smooths out (epistemic) noise associated with measurement.
- Potentially more interesting for real downstream applications, e.g. given temperature data at station locations *now*, what are predictions of temperatures elsewhere?

Overall, the DWD provides data collected from 501 stations at hourly intervals, with the closest stations being 3.73 km apart and a mean separation between stations of 19.37km. The full distribution of station separations is shown in Fig. 5.3.

[1]The time of day clearly affects temperatures via sun-based heating, but also through secondary effects such as coastal winds due to the different heat capacities of water and land.

The minimum separation is particularly important because it determines the smallest lengthscale over which the model can receive a signal during training.

In our experiments, we consider recent data between 2021-11-07 and 2023-05-10. These data do not overlap with the ERA5 training set so we avoid temporal leakage of ERA5 predictions into the ConvCNP. Significantly more historical data are available but we found this quantity of hourly data to be sufficient.

5.3 Auxilliary Data

To aid predictions, we provide the model with high-resolution elevation data (see Fig. 5.1, right) recorded by the NASA shuttle Radar Topography Mission (Farr et al., 2007). We also input the time of day, day of year, *and location information* (normalised latitude and longitude). This is significant because it breaks the translation equivariance, but it allows the model to learn localised temperature phenomena, such as differences between e.g. land/sea or forests/cities.

Vaughan et al. (2022) use a model architecture where the ConvCNP outputs elevation-agnostic feature-maps $\mathbf{h}$, and final predictions are made by combining $\mathbf{h}(\mathbf{x}_T)$ with high-res elevation data $\mathbf{e}_T$ through a multi-layer perceptron (MLP) ψ_θ:

$$m(\mathbf{x}_T),\ s(\mathbf{x}_T) = \psi_\theta(\mathbf{h}(\mathbf{x}_T), \mathbf{e}_T). \tag{5.1}$$

This allows the elevation data to be at an arbitrary resolution, without the ConvCNP needing to match that resolution (which is associated with larger feature maps and therefore slower computations). It comes at the disadvantage of the model only being able to see *pointwise* elevation data: the surrounding topography never enters the model, e.g. a plane at 1km elevation looks the same as the peak of a mountain at 1km elevation.

For temperature measurements, pointwise elevation data are sufficient[2], but for more complicated weather phenomena, such as precipitation, the surrounding elevation matters significantly. Motivated by these more complicated phenomena, we test a different architecture, where the elevation data enter at the ConvCNP encoder alongside the context temperature data.

Because the ConvCNP now needs a higher resolution to match that of the higher-res elevation data, we need to balance the trade-off between elevation data resolution and computational speed, as specified by the resolution of the ConvCNP (or points-per-unit (PPU)). Our choice of PPU is detailed in Sec. 5.5.

In future work (e.g. when modelling more complicated weather phenomena), a hybrid approach might be worth considering, where medium-resolution elevation data are passed through the ConvCNP, *and* high-res elevation data are injected via MLP after predictions are made. The model can then consider the rough topography surrounding the target locations cheaply, while still being provided with high-resolution elevation data.

[2]In hindsight, following (Vaughan et al., 2022) and injecting elevation data via MLP would've likely been the more appropriate design choice for our temperature downscaling experiments.

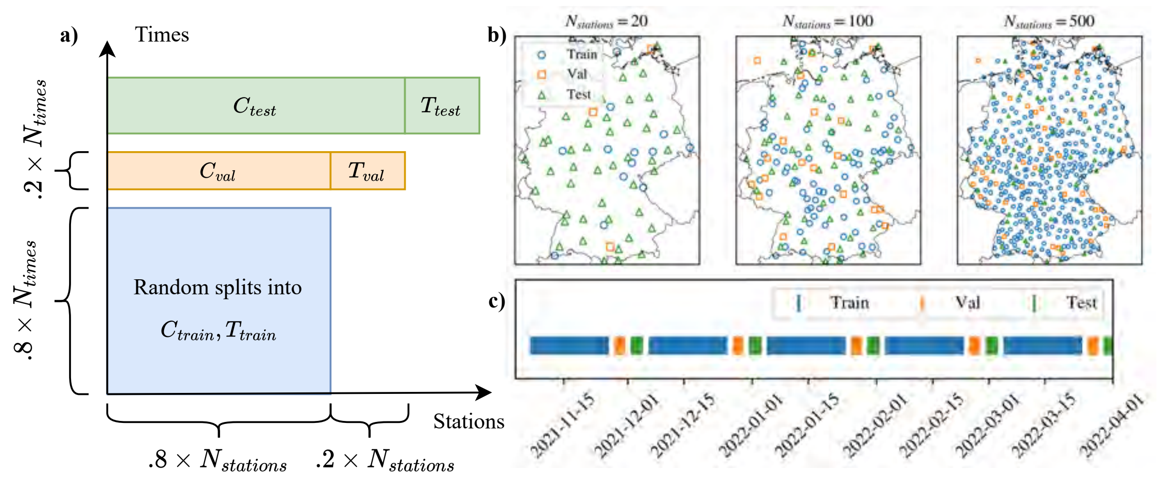


Figure 5.4: a) The N_{times} times and $N_{stations}$ stations each get split into 80% training and 20% validation data. For validation, we use training stations as context. For final testing, we use all available stations (train and val) to achieve the best results. The set of *test* times and stations is set aside at the beginning and is used for all evaluations. b) Train/Val/Test stations for different $N_{stations}$. c) Train/Val/Test dates are sampled throughout the available period with 2 days discarded between to avoid leakage.

5.4 Data Splitting

Spatiotemporal modelling makes splitting data significantly more complex than it is in most traditional machine learning domains. Generating training, validation and testing sets is not as simple as splitting all available data randomly. The most important problem is that the model can overfit both spatially and temporally – both of which can leak into the test/validation sets unnoticed unless care is taken.

We, therefore, use this section to outline our data-splitting procedure in detail, addressing this and other problems carefully.

The data-splitting process is further complicated by the fact that we're exploring different data splits to investigate different data-availability regimes.

5.4.1 Test Data

We split the available real data along two dimensions: time and stations. For consistent testing, we set aside the stations from the VALUE experimental protocol (Maraun et al., 2015). It provides a standardised set of experiments to evaluate downscaling methods and has a specific selection of stations in Europe, of which we select the 53 German stations. These stations cover a wide range of geographic features and are commonly used in downscaling experiments (Vaughan et al., 2022). Because some stations available in VALUE are covered by different copyright permissions than the DWD stations, they are not available to us. In those (9 out of 53 stations) cases, we instead choose the geographically closest DWD station. The furthest discrepancy from the VALUE stations is $\sim$ 18km, with most distances below 10 km. All VALUE stations (and substitutes) are shown in Fig. 5.3 (right).

5.4.2 Training and Validation Stations

The Train/Val stations are selected in a random order that we then keep fixed so that the stations in the $N_{stations} = 20$ experiments are a subset of the stations used for $N_{stations} > 20$ experiments, which ensures that no information is lost as we increase $N_{stations}$ (see Fig. 5.4 b)). We also investigated choosing stations that are as far apart from each other as possible but found that this significantly hurts what the model can learn, as shorter lengthscale signals are only featured for large $N_{stations}$ in this scenario.

5.4.3 Training and Validation Times

To avoid distribution shift between the Train/Val/Test tasks due to macroscale changes (e.g. climate change, el Niño/la Niña events etc.), we sample times throughout the available range, cycling between 19 days of training data, 2 days of validation data, 2.5 days of testing data, each separated by 2 days that we discard to avoid partial leakage due to correlated tasks, see Fig. 5.4 c). In this approach, we broadly follow Andrychowicz et al. (2023b), but we increase the number of discarded days from 1 to 2 to further reduce leakage. When we restrict ourselves (artificially) to a limited number of times, we select a random subset of times from the Train/Val pool that we keep fixed across experiments.

5.4.4 Generating Tasks

Once we have selected $N_{stations}, N_{times}$, we try to imitate what we would do for best model performance on downstream applications, with the limited numbers of stations and times available. In such a scenario, we want to maximise the model performance using all $N_{stations}$ available stations.

Given our restricted $N_{stations}$ and N_{times}, we follow this procedure (visualised in Fig. 5.4 a):

1. Split $N_{stations}$ into 80% training and 20% validation stations.
2. Split N_{times} into 80% training and 20% validation times.
3. For training, generate random subsets of context and target sets C_{train}, T_{train} only from the set of training stations and times. A single task is drawn as follows:
    (a) Select a random point in time t from the training times *without replacement* so that each t will be encountered equally often during training.
    (b) Draw a fraction $r \sim U(0, 1)$.
    (c) Denoting the number of observations at time t as $N_{stations}(t)$, we select a random subset of size $r^2 \times N_{stations}(t)$ as the *context set* C. The squaring of r makes sparser tasks more probable, which we find accelerated training for large values of $N_{stations}$.

   (d) Use the remaining stations as the target set T.

   (e) Note that this means a very large number of distinct (but related) tasks can be drawn from a single time t, as any combination of context stations is a “distinct” task.

4. For validation, use *all* available training stations as context C_{val} and all available validation stations as target T_{val} on unseen times from the validation times.

5. For testing, use *all* available stations, training *and* validation stations as context C_{test}, and the test stations (at test times) as targets T_{test}. This corresponds to the real-world scenario of using all available stations (train and val) for application. However, this *does* mean the model is tested in a regime that it has not encountered during training (a greater number of context stations).

5.4.5 Spatial or Temporal Validation

We also investigated whether or not the model tends to overfit spatially or temporally by qualitatively[3] comparing the validation loss during training to

1. Spatial validation loss, with stations $\subseteq$ validation stations, but times $\subseteq$ training times,

2. Temporal validation loss, with stations $\subseteq$ training stations, but times $\subseteq$ validation times.

If overfitting happened to be mainly temporal in nature, it would mean that we would not need a held-out validation set and could train on all $N_{stations}$ stations. However, we found that unless $N_{times} \lesssim N_{stations}$, (which would be very unusual in reality) overfitting was mainly spatial in nature.

Overall, we consider 3 numbers of stations: $N_{stations} \in \{20, 100, 500\}$[4] and 5 numbers of times $N_{times} \in \{16, 80, 400, 2000, 10000\}$[5] across our experiments to investigate different levels of data availability.

5.5 Experimental Deatils

Normalisation We normalise input data so that each of the two spatial coordinates is normalised to the range [0, 1]. Additionally, we normalise temperatures by subtracting their sample mean and scaling by their sample standard deviation. We save normalisation parameters during pre-training and use them during fine-tuning for consistency. This is performed using the *deepsensor* package (Andersson, 2023).

[3]By comparing if the validation loss (held out stations and times) had minima that aligned with spatial validation loss or temporal validation loss.

[4]Note that because we only have 501 stations with 53 reserved for testing, the $N_{stations} = 500$ setting actually only incorporates 448 stations for training and validation.

[5]It is unlikely that any weather station has only collected 16 datapoints in reality, but we include this regime for generality and potentially analogous applications in other domains.

Model Hyperparameters We use the model architecture described in Sec. 2.2 with 6 layers (down and up) in the U-Net, of 96 channels each. We choose a resolution of 200 Points Per Unit[6] (PPU) for the internal gridded representation and a corresponding encoder and decoder lengthscale $\ell_E = \ell_D = 1/200$, which allows us to comfortably resolve the smallest station separation (see Fig. 5.3) in normalised space. This resolution is too high for the coarsely-gridded ERA5 data during pre-training but allows for re-using the model without any transformations. In total, our model has 3.8 million parameters, out of which 3284 (0.08%) are FiLM parameters.

This approach is slightly inefficient: we pre-train the model on an unnecessarily high resolution. In Future work, methods of training on the lower resolution of the simulator data, and then transferring parameters to a higher-resolution version of the model for fine-tuning could be explored.

This could, for example, be performed by appending higher-resolution layers at the top of each of the U-Net legs when fine-tuning on higher-resolution data. These filters are initially parameterised in such a way that they average the higher-resolution data to the original pre-training resolution[7]. This would preserve ρ_θ while then allowing fine-tuning at higher resolutions.

Optimisation We use the Adam optimiser (Kingma and Ba, 2014), with a learning rate of 1×10^{-4} during pre-training and 3×10^{-5} during fine-tuning, both of which were found via grid search as in Sec. 4.2. We use a batch size of 16 throughout.

Because each point in time yields a very large combination of context and target-set combinations (which are related but distinct) an "epoch" in the traditional sense is far too large to be useful. We instead define an epoch as 200 batches (i.e. $200 \times 16 = 3200$ tasks) during pre-training.

We anneal the learning rate by a factor of 3 if the validation loss stalls for more than 8 epochs, which helps for convergence at the end of training. We stop after 20 epochs without improvement.

During fine-tuning, we define epochs to be smaller, each consisting of 25 batches, to monitor validation losses more frequently. This definition of epoch also allows for consistency across different N_{times} and $N_{stations}$. During fine-tuning, we stop after 30 such "epochs" without improvement.

Finetuning Approaches We investigate FiLM adaptation and global fine-tuning in detail. We also run preliminary experiments with fine-tuning just short-range U-Net layers, or just short-range U-Net layers and FiLM layers, but quickly discard these approaches based on poor initial results.

5.6 Baselines

In all Sim2Real experiments, we consider two baselines: "sim" only and "real" only.

[6]Note that this means 200×200 grid-points per 1×1 unit square

[7]Small amounts of noise need to be added to the initial parameters to break symmetry.

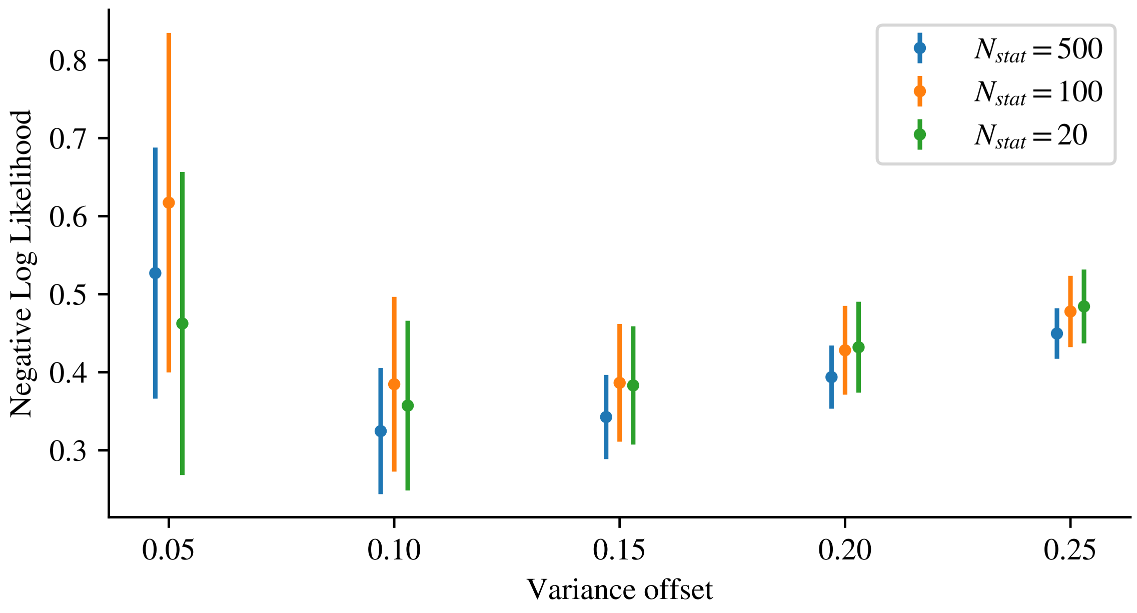


Figure 5.5: Adding some variance reduced the impact of overconfidence on short scales, leading to significantly improved NLLs, and serves as a fairer baseline across station densities. Here N_{stat} is the number of stations used as context during evaluation.

5.6.1 Real Only

In this setup, we skip the pre-training part and directly train the ConvCNP (from random initialisation) on the available real data. Particularly in sparse-data regimes, this method is expected to underperform, as the ConvCNP can be data-hungry to train.

5.6.2 Sim Only

In this baseline, we train the ConvCNP on simulator data and apply it to real data without fine-tuning. We found that the mean predictions were generally quite strong when making real predictions using such sim-only models, but some predicted variances σ_i^2 were close to 0 where they should not have been (particularly on short lengthscales $< \ell_{grid}$, which are not featured in the simulator). This lead to very high negative log-likelihoods $\mathcal{L}_{sim} \gg 100$ (for reference, our best models achieve $\mathcal{L} \approx 0$).

To make this baseline achieve more competitive Negative Log-Likelihood (NLL)s, we shift all predicted variances by an offset δ_{σ^2}:

$$\sigma_i^2 \leftarrow \sigma_i^2 + \delta_{\sigma^2}, \quad \forall\, i = 1 \dots N_T. \tag{5.2}$$

We determine the value of $\delta_{\sigma^2} \approx 0.1$ for $N_{stations} = 500$ and $\delta_{\sigma^2} \approx 0.15$ for $N_{stations} \in \{20, 100\}$ by coarse 1d grid search[8] on a very small number of (training) samples

[8]This involved changing the code of an external library on our local machine. We did not consider it a good use of time to transfer these changes to the high-performance computers to reduce errorbars of this plot, or to employ grid-search over both an additive offset and a multiplicative factor.

($N_{times} = 32$) (see Fig. 5.5). This is technically Sim2Real transfer, but because we are tuning a single parameter on a very restricted dataset, we consider it sim-only for the purposes of comparing models.

5.6.3 Variance Offset for Sparse Stations

In the sparse-station regime $N_{stations} \leq 100$, and the sparse-task regime $N_{stations} \leq 400$ we observed a similar pattern of overconfidence on very short lengthscales. In the sparse-station regime, this happens because these lengthscales are also infrequently (if at all) covered by the fine-tuning data. In the sparse-task regime, early stopping means that the model is still very similar to the simulator (and therefore also overconfident on short lengthscales).

We, therefore, give these fine-tuned models the same benefit of adding a variance of 0.15 to have a fairer comparison between Sim2Real and variance-offset sim-only[9].

5.7 Results

Note: Throughout this section, we are only able to present results from single runs, with fixed "sim" starting parameters, fixed training and validation times fixed training stations and crucially, fixed validation stations (see Sec. 5.4) because of computational constraints. For this reason, our results do not include error bars and are associated with significant noise: Because there is a mismatch between the testing and the validation tasks, the random noise associated with which stations are used during training and for validation is large. Particularly in the $N_{stations} = 20$ (and to a lesser extent the $N_{stations} = 100$) case, the very small number of held-out validation stations can be very mismatched to the test-stations, adding noise through bad early-stopping. Particularly the NLL is sensitive to this noise.

In Fig. 5.6, we compare the performance of our Sim2Real transferred model to that trained solely (from random initialisation) on available real data. Clearly, the initialisation at simulator-pre-trained parameters is very helpful for training the model in all but the largest real data regime ($N_{times} = 10000$), showing the utility of Sim2Real in low-data regimes.

Fig. 5.7 shows how the Sim2Real fine-tuned models compare to the *sim-only* baseline. Interestingly, we see qualitatively different behaviours in different data availability regimes:

- In the sparse-station setup $N_{stations} = 20$, the model is unable to improve through Sim2Real, no matter the quantity of tasks (N_{times}). Note: the fine-tuned model can get worse than the sim-only baseline from which it starts because the validation set is itself small and might not reflect the test set well.
- In the dense-station setup $N_{stations} = 500$, the model does improve significantly, even given very small amounts of real data (e.g. $N_{times} = 16$).

[9]Without this adjustment, the variance-offset sim-only model achieved much better NLLs than the not-offset fine-tuned models, simply because of short-range overconfidence. This did not align with MAE.

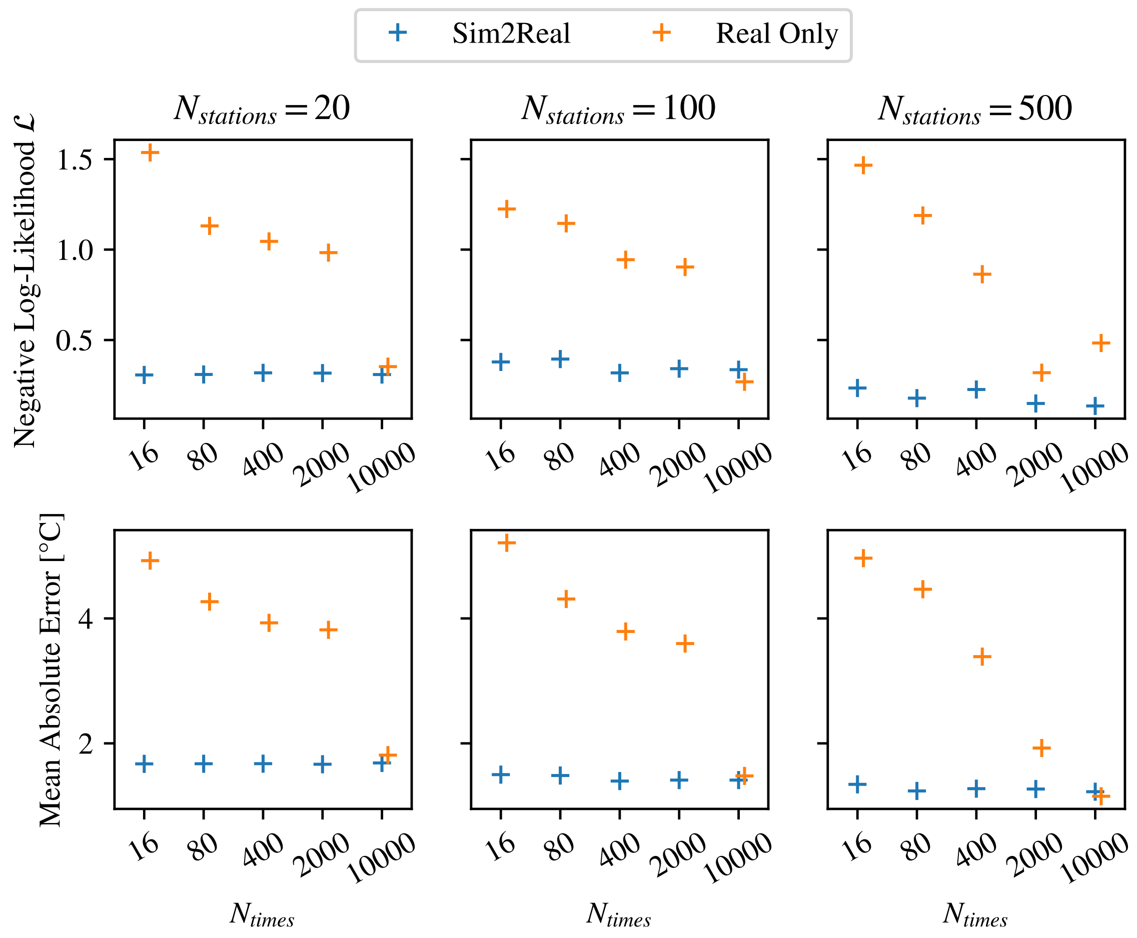


Figure 5.6: pre-training a model on simulator data significantly aids performance compared to starting from random initialisation. Only with large amounts of real data do the performances become comparable. Note: Fine-tuned model performances (blue) do change with additional training data, but given the scale of the y-axis, this is better shown in Fig. 5.7.

- In the middling station density $N_{stations} = 100$, the model does improve slightly given enough tasks $N_{tasks} \gtrsim 400$.

These results show that Sim2Real is not always useful, especially when the real data covers a much smaller density of context and target points than the simulator data. However, given even a modestly sized real dataset, Sim2Real can yield significant model improvements and, as we show in the following sections, large improvements in qualitative behaviour.

5.7.1 FiLM

We also attempted initial experiments using FiLM adaptation, but it did not outperform global fine-tuning in any regime. For sparse stations, it performed marginally

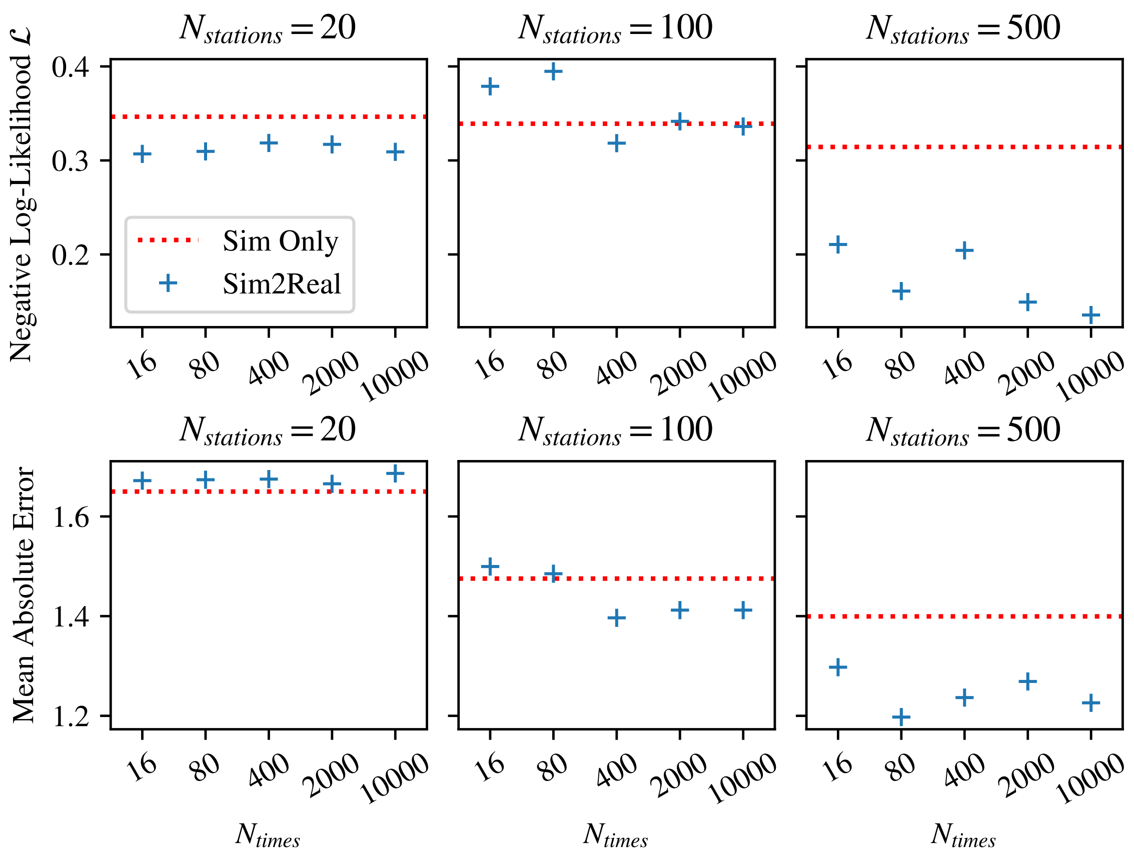


Figure 5.7: Fine-tuning is only useful when the fine-tuning data is sufficiently different from the simulator. In the $N_{stations} = 20$ regime, the model is unable to improve beyond the simulator, even after many training tasks. In higher data-availability regimes, fine-tuning leads to clear improvements.

worse than global fine-tuning, and for dense stations significantly worse. We believe that the explanation is two-fold:

1. This task, even in the sparse regime, involves changing lengthscales (e.g. smaller radii of "certainty" around context observations), which FiLM adaptation cannot do (see Sec. 4).

2. The rings of poor prediction around context observations, which we call "artefacts" (see Sec. 5.10) are difficult to reduce without changing the convolutional filters.

While it would be interesting (and in future work worthwhile) to investigate these hypotheses, we prioritised other work under the time constraints.

We now investigate the different ways in which Sim2Real changes/improves the model.

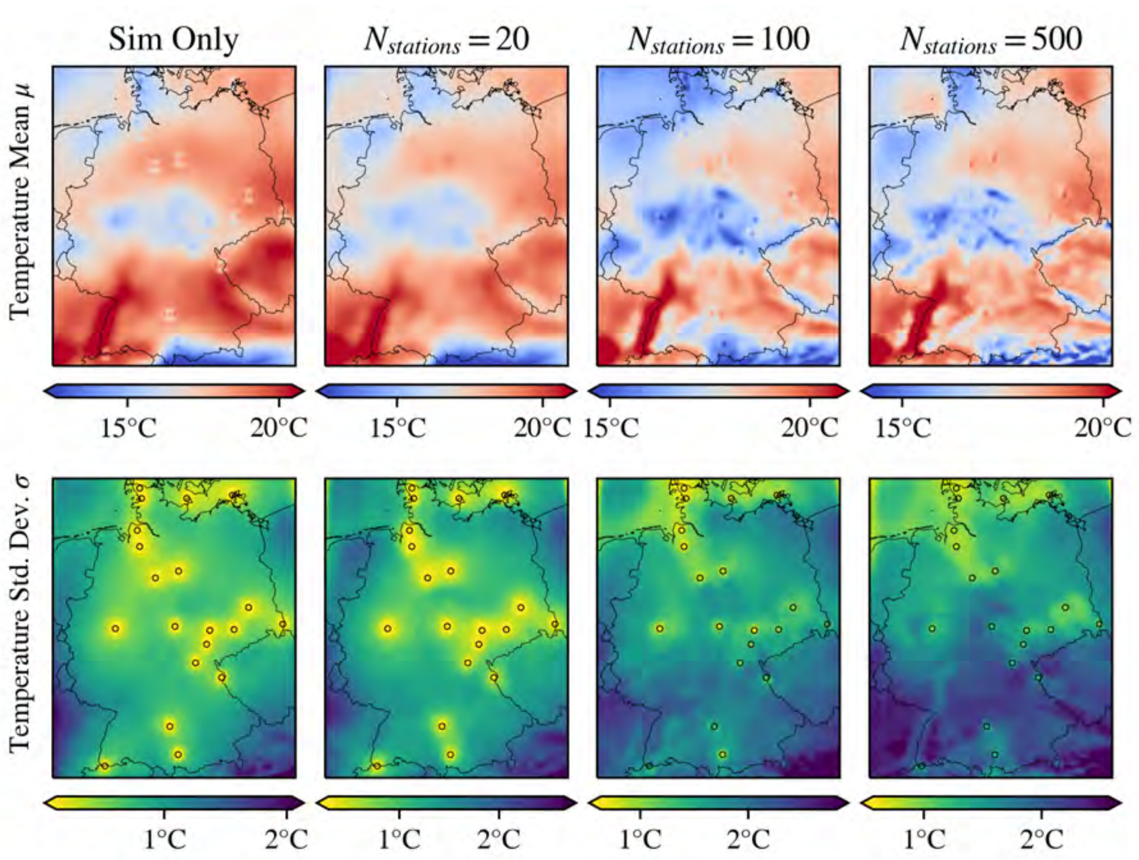


Figure 5.8: As we increase the number of stations shown in the fine-tuning process, the model makes predictions on increasingly short lengthscales. The predicted standard deviation σ around drops off significantly faster around the $N_C = 20$ context stations (black circles) with fine-tuning. All models are fine-tuned on $N_{times} = 10000$, with fewer training examples showing similar, but less pronounced, effects.

5.8 Sim2Real Learns High-Frequency Features

One of the main contributors to the Sim2Real gap is the fact that learnable patterns are restricted to lengthscales above the grid-spacing ℓ^{sim}_{min}.

Fine-tuning the model on real data shortens ℓ_{min} to the shortest inter-station separation, ℓ^{real}_{min}, which is a factor of 5 smaller than ℓ^{sim}_{min} for the densest station setting of $N_{stations} = 500$. This leads to higher-frequency features in the model's predictions, modelling shorter-lengthscale weather phenomena, which visually looks like an increased resolution. This is shown in Fig. 5.8.

The visual artefacts in μ surrounding context observations (visible clearly in the top row) are explained and thoroughly addressed in Sec. 5.10.

The ConvCNP only pre-trained on simulator data models standard deviations σ homogeneously across the country: where there are context observations, the model predicts a high degree of certainty around that observation (bottom row, left in Fig. 5.8).

As we increase $N_{stations}$ during fine-tuning, the model learns to better evaluate how valuable an observation is: In regions with multiple consistent measurements, the observations reduce uncertainty in a wide region (e.g. Northern Germany in Fig. 5.8, bottom row, right), if observations do not align, or are in a region of rapid change, they do not provide much certainty (e.g. central Germany).

On a more macroscopic scale (away from context stations), uncertainties in regions of rapidly changing temperatures (central Germany in Fig. 5.8) are higher, and lower in more homogenous regions (northern Germany). Uncertainties are also higher in mountainous regions (where temperatures can change dramatically). Overall, predicted uncertainties are higher. All of these changes become more pronounced with more fine-tuning stations.

The dramatic changes in predicted uncertainties have important implications for downstream applications. For example, predicted uncertainties have been used to propose new sensor locations: Andersson et al. (2023) consider sensor placement in Antarctica, where station locations are sparse and lengthscales longer, so the ERA5 baseline is a "good enough" proxy for real observations[10]. However, in Germany, the method used by the authors proposes significantly different sensor-placement locations with and without fine-tuning, as we show in Sec. 5.12.

5.9 Sim2Real Learns Elevation Dependences

The coarseness of ERA5 also leads to a limited ability of the sim-only ConvCNP to incorporate the high-resolution auxiliary elevation data: Because simulations at grid points represent a blurry "average" over the grid cell, the correlation between elevation and temperature too is blurry. This restricts what the ConvCNP can learn. The top of Fig. 5.9 shows: While the model has a coarse understanding that regions of higher elevation are associated with colder temperatures, it is unable to model rapidly changing temperatures in mountainous regions. Note that there are no context observations in this region, the model relies on auxiliary data to make heterogeneous predictions.

As we fine-tune the model on real data, which consist of *point measurements*, the model is able to learn higher resolution features (bottom of Fig. 5.9), gaining a high-res understanding of the effect elevation has on predictions. For this, the number of stations $N_{stations}$ appears to be vastly more significant than the number of observations N_{times}.

5.10 Discrete $\mathbf{x}_T, \mathbf{x}_C$ Lead to Shortscale Artefacts

A limitation associated with distilling a gridded simulator into a ConvCNP is the fact that the ConvCNP never receives a training signal shorter than the grid spacing of

[10]This is supported by the similarity in predictions between the sim-only model and the $N_{stations} = 20$ model and the results in Fig. 5.7. However, the authors' approach would've potentially benefitted from an offset variance as described in Sec. 5.6.2.

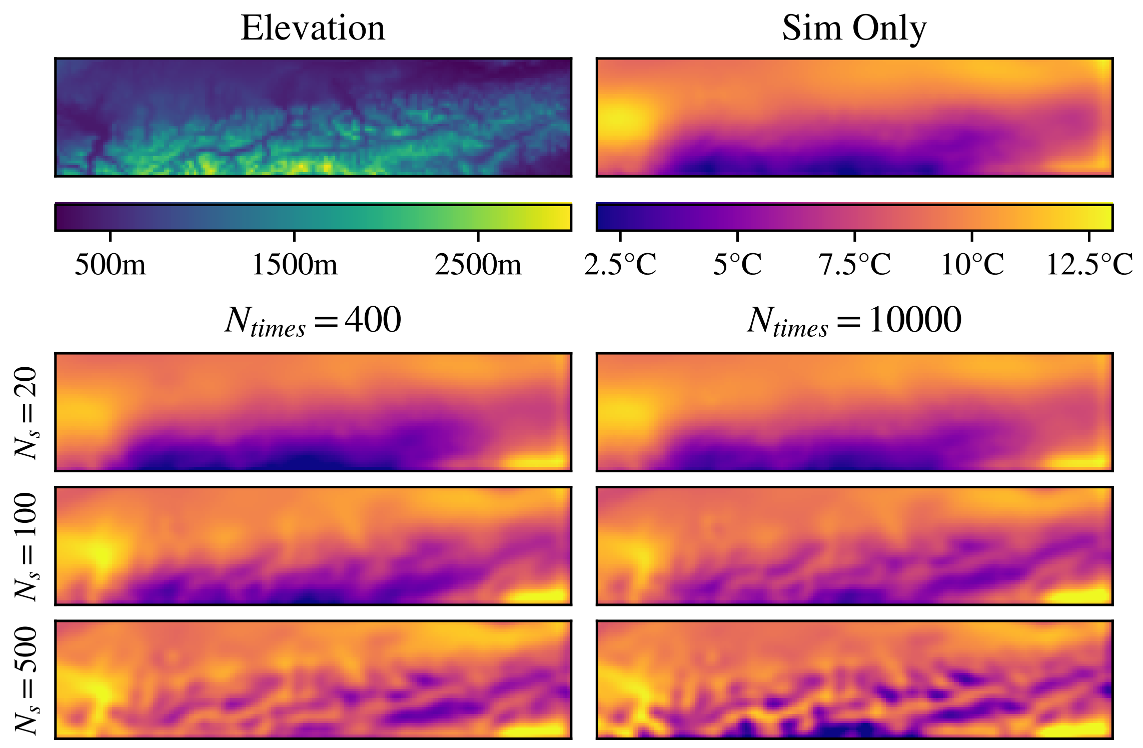


Figure 5.9: High-frequency patterns are learned over regions where elevation rapidly changes during the fine-tuning phase. For this, increasing the number of real stations (here denoted N_s for brevity) is more effective than increasing the number of training tasks.

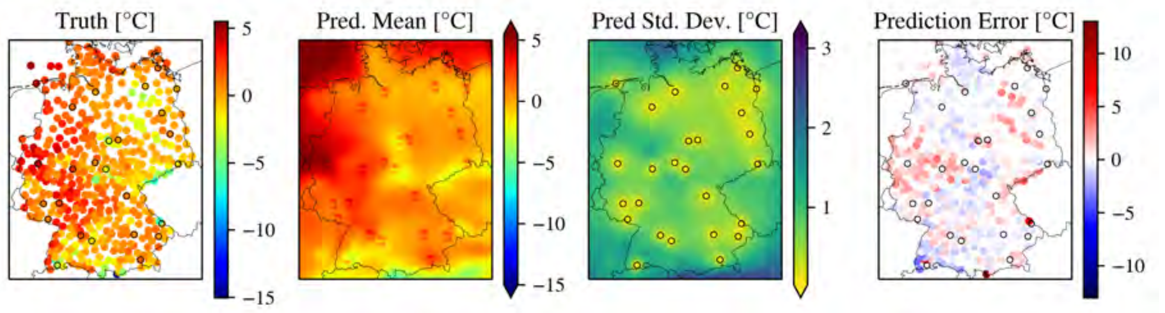


Figure 5.10: A ConvCNP which has been pre-trained on ERA5 simulator data and not yet fine-tuned, predicting temperature using real context measurements. Immediately around context observations, the model predicts temperatures that clearly do not align with surrounding predictions.

the simulator data, ℓ_{min}. Attempting to predict at higher resolutions than ℓ_{min} can lead to clear visual artefacts (see Fig. 5.10) around context observations.

Artefacts are instances of the model failing to smoothly fit context observations into its wider predictions, so a good way to distinguish between valid high-frequency features and invalid context-point-induced artefacts is to consider whether or not we can guess the context locations X_C based on *just the predictive mean*. In the predictive mean plot of Fig. 5.10, for example, it is easy to see context locations by the discolouration (red smears) around each context point: The model is unable to integrate the context observations smoothly. The bottom row of Fig. 5.11 shows similar artefacts. In contrast, the top row of Fig. 5.11 likely shows "valid" high-frequency features: while there are rapid temperature changes immediately around context points, they are integrated cohesively into the prediction, and there are also rapid temperature changes *away from the context points*.

These artefacts are often inconsistent with both the context observation itself and surrounding predictions and occur because the model's loss is never punished for predicting them: because the closest possible target location is ℓ_{min} away from any context point, the model's predictions on lengthscales $\ell < \ell_{min}$ has no effect on the loss. The model has not encountered signals of $\ell < \ell_{min}$ in the data and is not endowed with any prior knowledge of temperatures on short scales, it is also unable to extrapolate to shorter lengthscales from the available $\ell > \ell_{min}$ data.

There are a number of ways of removing these artefacts, most of which correspond to injecting prior knowledge about how temperature should behave into the model or the data. For example, we could

- Set the decoder lengthscales ℓ_D to the grid-spacing. This corresponds to asserting that the temperature should be distributed as a Gaussian bump around the context observation (or whatever other decoder function ψ_D is used, see Sec. 2.2).
- Low-pass filter model predictions for frequencies greater than $1/\ell_{min}$.
- Only predict on a resolution of ℓ_{min} (similar to filtering).
- Apply noise to the position of context and target points.
- Sample training context and target points continuously and interpolate the temperature between grid-points.

We experiment with the latter option using linear interpolation and (as expected), the artefacts disappear.

However, this approach is limited because the model grows overconfident on lengthscales $\ell < \ell_{min}$, as it learns to linearly interpolate. We could additionally add noise (perhaps related to the distance from the grid point), and perhaps shift the temperature linearly according to surrounding elevation, but this highlights the key issue with these approaches: we are effectively constructing a short-range model based on prior knowledge to generate data to train the ConvCNP. This is domain-specific, knowledge reliant and therefore not in the spirit of this project.

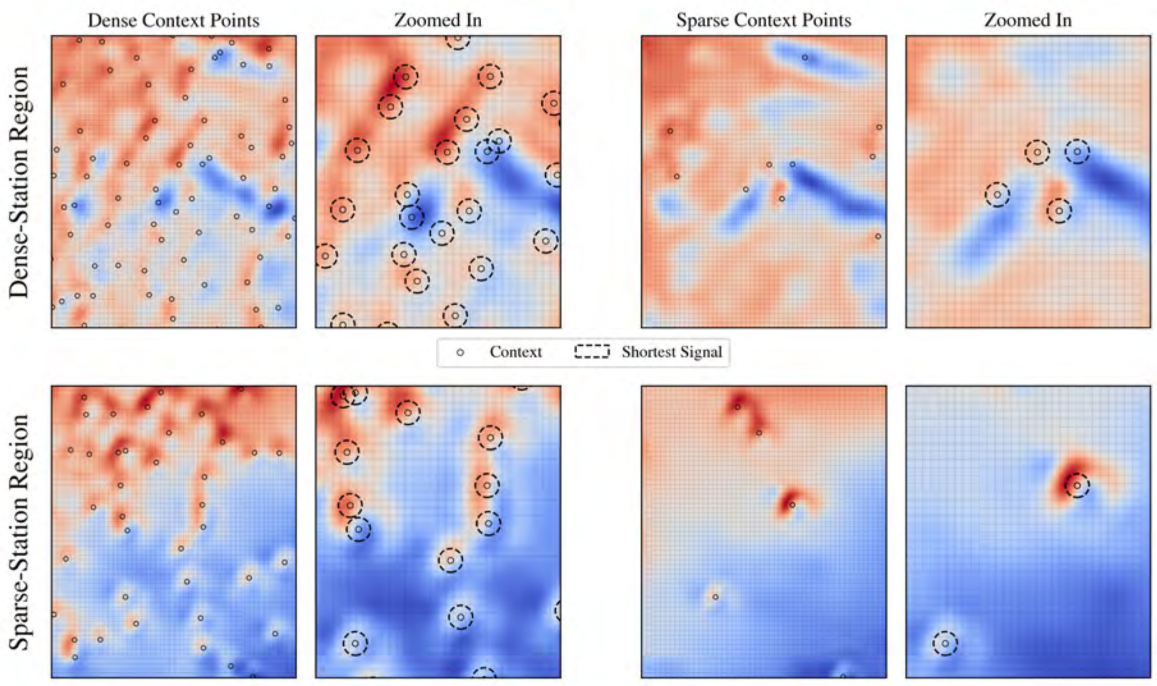


Figure 5.11: Artefacts for the $N_{stations} = 500, N_{times} = 10000$ fine-tuned model. The grid represents the internal grid of the ConvCNP and is on the same order as the smallest inter-station separation (dotted circles). Artefacts appear mostly in regions of Germany that have fewer stations (bottom row), while in denser regions artefacts are less pronounced (top row). Temperature scales are hidden to avoid clutter, but artefacts are on the order of $\sim 1°\text{C}$.

Any of the approaches relying on smoothing out short-lengthscale features lose any detail associated with other context data, such as the high-res elevation data. In Sec. 5.10.2, we describe a method that neither filters high-frequency outputs nor relies on prior knowledge.

5.10.1 Artefacts in Sim2Real

Until now, we have only described artefacts due to the fixed grid-spacing ℓ_{min} during the pre-training phase of Sim2Real.

The issue of artefacts, however, persists also through the fine-tuning phase on real station data, with the new shortest signal ℓ_{min} given roughly by the smallest inter-station separation within the training stations.

Across domains, spatiotemporal models will not receive any signals on lengthscales $\ell < \ell_{min}$ from the data, so unless context and target points are continuously sampled ($\ell_{min} = 0$), artefacts might appear.

As shown in Fig. 5.3, $\ell_{min} \approx 4$km when using all stations for training (and slightly above for fewer stations), which is far smaller than the grid-spacing during pre-training (∼20km). Accordingly, artefacts reduce in scale with fine-tuning (see Fig. 5.8).

Fig. 5.11 yields the following interesting observations, which we verify by checking

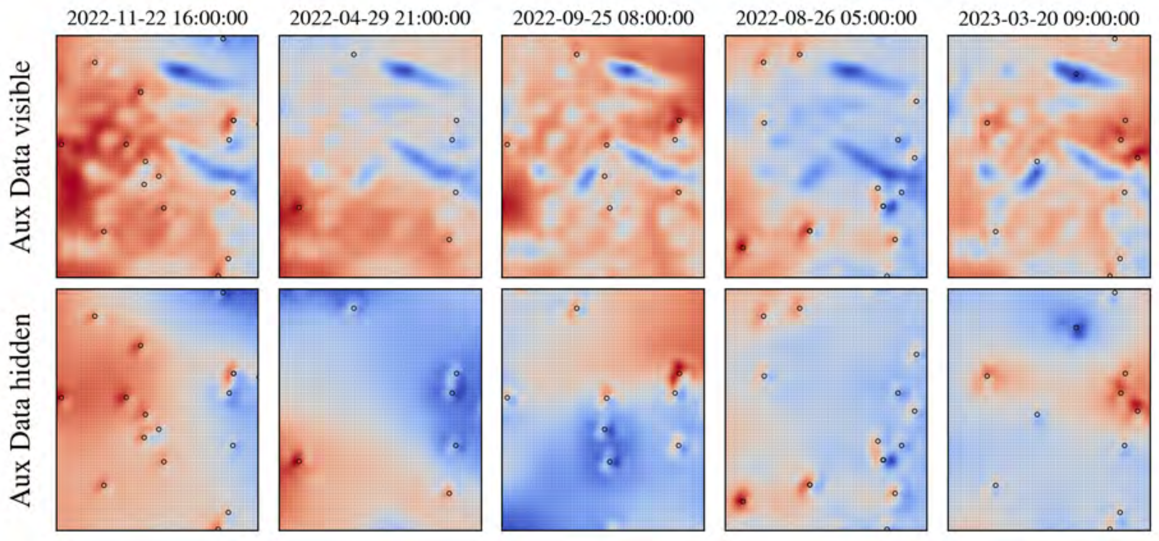


Figure 5.12: Hiding translation-equivariance-breaking auxiliary data appears to increase artefacts around context points. Note that every plot uses its own temperature scale for improved visibility. Structural similarity across dates in the top row is due to elevation.

a number of other independent samples:

1. Artefacts follow a similar pattern at multiple context stations, e.g. a warmer semi-circle to the north of context points, with a colder patch just south. This might indicate a strong contribution of individual convolutional filters.

2. Artefacts can be larger than ℓ_{min} (see bottom row, dashed circles are of radius ℓ_{min}).

3. Artefacts appear to be *region-dependent.* In regions of Germany that feature a short station-spacing, i.e. a high density of stations (such as the Rhine-Main Region in central Germany), artefacts seem less pronounced than in regions with fewer stations (such as the north-east of Germany)[11].

The latter two observations are particularly interesting: We hypothesise that the model learns, based on the equivariance-breaking auxiliary data (elevation data, x, y location map), that artefacts of scales $> l_{min}$ do not have an effect in the sparser regions of Germany, promoting stronger artefacts in these regions.

As preliminary evidence that does not involve re-training, we compare what happens to predictions in the station-dense regions of Germany when we hide the translation-equivariance-breaking auxiliary data, setting $x = y = 0.5$ and elevation $= 1$ (in normalised units). The model now no longer knows that it is in a high-density region and might produce stronger artefacts. In line with this hypothesis, artefacts become significantly worse (see Fig. 5.12).

[11]It is much harder to guess context locations based on predictions in the top row (dense) than the bottom row (sparse) of Fig. 5.11.

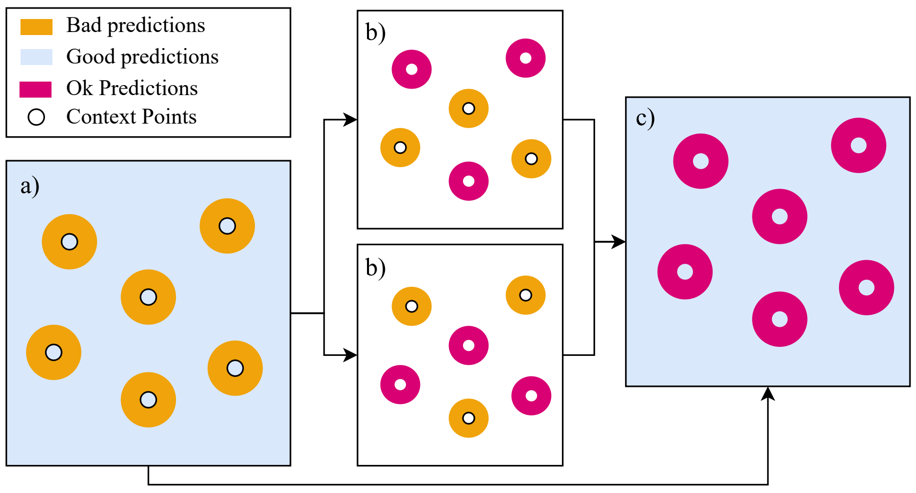


Figure 5.13: Stitching together different predictions can improve artefacts. a) Around our N_C context observations, we have a ring (orange) of bad predictions. Immediately at the context observations and outside the artefacts, we have good predictions. b) We split the N_C context observations into $k = 2$ disjoint sets, and obtain predictions around the held-out context stations (pink), where there would otherwise be artefacts. c) The different $k + 1 = 3$ predictions are stitched together to create a final prediction.

This experiment is not *proof* that our hypothesis holds, especially because the model has never seen constant x, y and elevation channels and might behave strangely because of it, but does provide some evidence towards it[12].

We propose, as future work, to investigate artefacts further:

- Train a less powerful model without any auxiliary data to investigate the region-dependence.
- Sample context points on a discrete grid in the toy experiment (Sec. 4) and investigate the behaviour of artefacts in a simpler setting.

A counter-example is given by the 4th column in Fig. 5.12 (2022-08-26), where artefacts appear visible even when auxiliary data are shown, so these hypotheses should be viewed with caution.

5.10.2 Artefact Stitching

We develop an approach to overcoming artefacts that does not rely on any prior knowledge (but comes with other limitations). We depict this approach in Fig. 5.13:

1. First, a prediction is made using all N_C context points. This prediction should be of high quality everywhere except in the artefacts.

[12]Especially as predictions away from context stations seem roughly OK even when auxiliary data are hidden.

2. Next, the available context points are split into k distinct sets. Holding one of the k sets out at each time, we make k predictions (each holding out one kth of the available context points). Each of these predictions should be able to predict (based on longer lengthscale correlations) the temperature within the artefacts of the held-out context stations. Because we do not use all N_C context observations, this prediction will (on average) be worse (and different) than for regions without artefacts.

3. Finally, the predictions are stitched together (by discarding regions covered by artefacts) to achieve a better overall prediction.

This method relies on the fact that the model *has* learned how to make medium-range predictions, which we use to replace the short-range predictions that cause artefacts. Note that we have not had to inject any short-lengthscale knowledge into the model (or data) to achieve this improvement and that the method is domain-agnostic.

This method is of course not perfect either: besides the $(k+1)$-fold increase in computational cost, there will still be visual discontinuities where we stitch together the k predictions, and we must define a method by which to exclude regions. One method is cutting out a circle of radius r around any context stations. However, this adds another hyperparameter and relies on the assumption that artefacts are of a homogeneous radius, which might not always be the case (see Sec. 5.10.1).

Due to time constraints[13], we only generate preliminary results, using the model fine-tuned in the medium-data regime $N_{stations} = 100, N_{times} = 400$.

Here, we find the NLL $\mathcal{L}$ improving[14] significantly with a small increase improvement in MAE:

	Ordinary Prediction	**Stitched Prediction**
NLL	1.17 ± 0.09	0.17 ± 0.03
MAE	1.34°C ± 0.02°C	1.33°C ± 0.02°C

How much of this improvement is associated with reducing artefacts and how much with simply increasing uncertainties in the stitched regions is for now unclear.

The improvement in prediction mean (shown by MAE) is small and not statistically significant given the limited number of test tasks we were able to compute on our local machine.

It is, however, a promising preliminary result, motivating future work to explore the method further.

5.10.3 Artefact-Free ConvCNPs?

Note: Some of the ideas and experiments presented in this section are those of Tom Andersson, one of my supervisors. I attempt to be clear about which observations were

[13]We had to change code in an external library to change the prediction method, and making these changes on the high-performance computers would've involved more work than we have time for.

[14]Because the test stations are unseen during training time, some test stations lie in artefact-prone regions affected by this method.

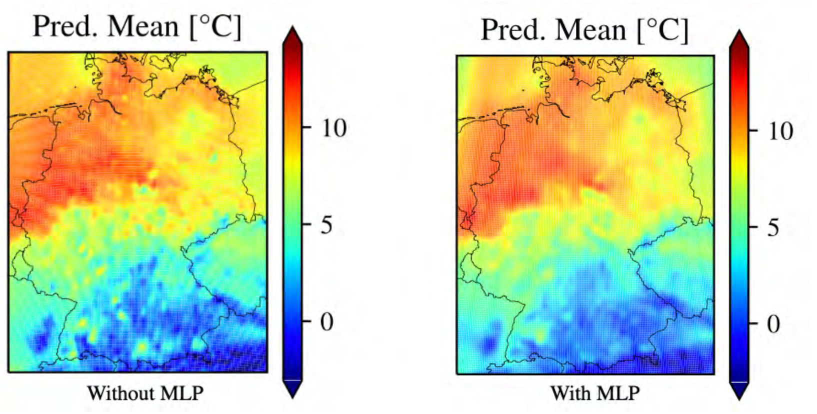


Figure 5.14: Adding MLP layers after the U-Net significantly reduces the visibility of artefacts.

made by him and which by myself.

Interestingly, my supervisor found that artefacts disappear when using the MLP-based architecture described in Sec. 5.3. This aligns with the work by others using this architecture, such as Vaughan et al. (2022); Markou et al. (2021b), who also do not report any artefacts on short lengthscales.

My supervisor hypothesized that the artefacts present in the context encoding are transmitted, unmitigated, through the residual connections between the different levels of the U-Net (see Fig. 3.1). Adding the MLP ψ_θ at the end could help smooth out these artefacts in a way the single linear layer that our model uses (see Fig. 3.1), can not.

To investigate this, I added an MLP (3×128 channels) between the U-Net and final classifications in a similar manner, but without passing elevation data $\mathbf{e}$. As shown in Fig. 5.14, artefacts are significantly reduced, supporting my supervisor's hypothesis. Interestingly, however, they are still somewhat visible, implying that the addition of elevation data itself might also help reduce the artefacts. To confirm this observation[15], my supervisor replaced $\mathbf{e}_T$ with constant features, confirming that the artefacts reappeared.

A more thorough investigation in future work is promising.

5.11 Catastrophic Forgetting

A problem with Sim2Real is *catastrophic forgetting*. As the data that the model is shown changes (from "sim" to "real"), the parameters of the model are adjusted to minimise the NLL on the real data. In doing so, the model can forget certain aspects of the "sim" data that might be useful for downstream tasks. For example, when pre-training on the $\sim$ 600 ERA5 context points and then finetuning on a small number of stations $N_{stations} = 20$, weather phenomena or short-scale behaviour could be lost, negatively affecting test-set performance.

[15]And to exclude potential bugs in my implementation

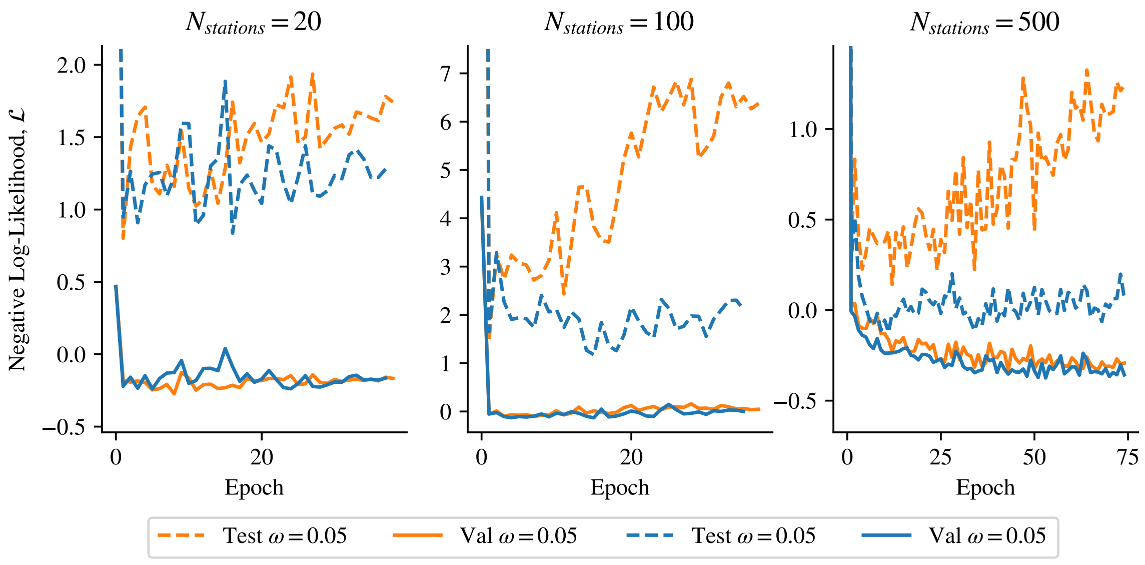


Figure 5.15: Injecting small amounts of simulator data has no effect on validation accuracy but *lowers* test performance across station densities. This experiment was run at $N_{times} = 10000$ to disentangle any potential *forgetting* from *overfitting*. Test loss is more volatile because we use a smaller number of tasks to avoid slowing down training.

Even in the densest $N_{stations} = 500$ case, no stations provide observations over the north sea. This will not affect test performance (because test stations too are land-based), but predictions might suffer.

We, therefore, attempt to counteract catastrophic forgetting by injecting a small fraction ω of simulator data into the *finetuning* process, hoping that the model closes the Sim2Real gap in the regime covered by real data while retaining what it has learned on simulator data. However, we found that injecting any amount of simulator data into the fine-tuning samples makes test-set performance worse as shown in Fig. 5.15.

To understand this, we refer back to Sec. 5.4: during testing, we use all $N_{stations}$ available stations to make predictions, which includes validation stations. During training, the model only ever sees $0.8 \times N_{stations}$, the fraction reserved for training. It has not seen the larger number of stations during training and therefore is expected to perform worse[16].

If we now inject ERA5 data into the fine-tuning process, it *does* see the denser station regime, within those ERA5-injected tasks. We hypothesise that this small number of injected tasks dominates the testing regime (of $> 0.8 \times N_{stations}$), leading to worse performance.

We also considered different ways of injecting simulator data into the fine-tuning process, such as merging real observations and simulator observations *within the same context set* to increase the density of "stations". However, because this method is domain-specific and not generally applicable, we choose not to pursue it further.

In the following final section, we investigate the impact of Sim2Real on the down-

[16]Only using the $0.8 \times N_{stations}$ training stations during testing achieves even lower performances.

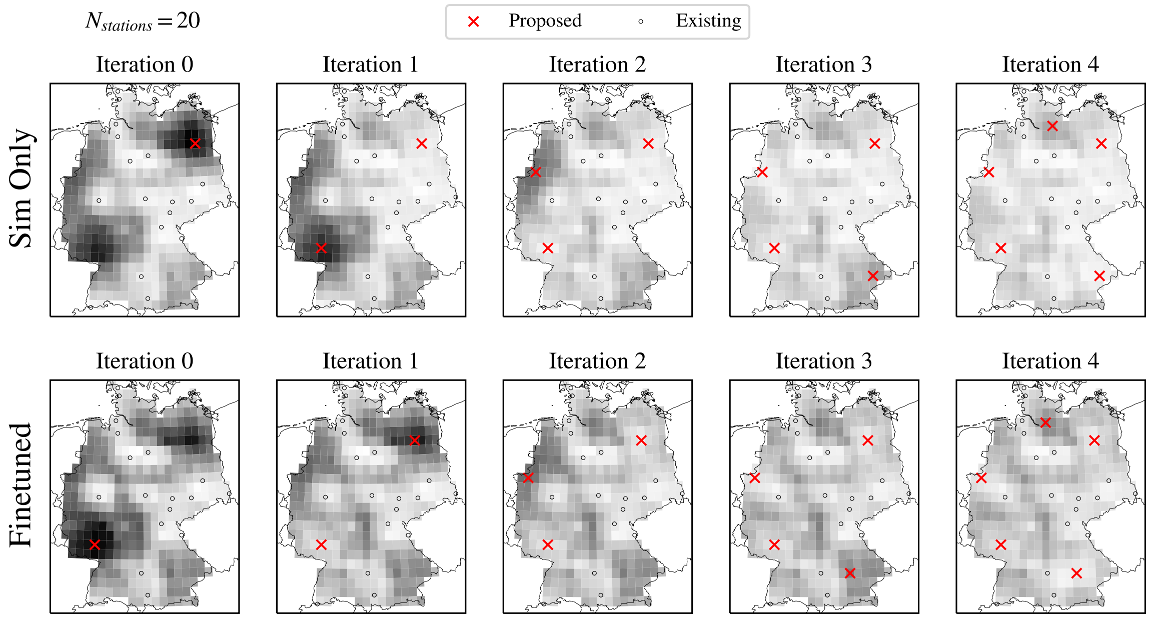


Figure 5.16: Fine-tuning the model on 20 stations does not significantly change the evaluated auxiliary function, nor the proposed station placements.

stream task of *active learning*.

5.12 Active Learning

ConvCNPs have been used for *active learning*, where their predictive uncertainties can be used to inform future decisions: Andersson et al. (2023) use a ConvCNP trained on ERA5 data over Antarctica and then apply the model to real observations to make informed decisions about where to place *new* weather stations (i.e. new context points). Employing Sim2Real in this application has the potential to significantly improve these already impressive results. A complete quantitative comparison is out of scope for this report, but we qualitatively investigate how different the placement locations of new weather stations would be if Sim2Real was employed, in our usual setting of Germany.

This process works by computing an *acquisition function* Γ, representing a metric, to evaluate search locations $\mathbf{x}_i \in S$:

$$\Gamma(\mathbf{x}_i) = \frac{1}{N_{times}} \sum_{j=1}^{N_{times}} \tilde{\Gamma}(\mathbf{x}_i, t_j) \tag{5.3}$$

averaged temporally over multiple tasks at times t_j.

New sensor placement locations $\mathbf{x}^*$ are then proposed according to:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}_i \in S} \Gamma(\mathbf{x}_i). \tag{5.4}$$

By adding $\mathbf{x}^*$ into the context locations (as if we had built a sensor there), and setting the observation value to the predicted mean $y^* = \mu_\theta(\mathbf{x}^*|C)$, we can iteratively repeat the process to (greedily) place multiple sensors.

In their paper, and the associated code-base *deepsensor* (Andersson, 2023), the authors provide a range of acquisition functions. Simple acquisition functions use the model's standard deviation as the acquisition function, i.e. $\Gamma(\mathbf{x}_i) = s(\mathbf{x}_i)$ (which would suggest sensor placement at the search points of highest uncertainty, averaged temporally over a number of tasks).

More complex acquisition functions compute information-theoretic metrics such as the mutual information between the model's prediction and a hypothetical sensor observation at $\mathbf{x}_i$ (set to the model's predicted mean at $\mathbf{x}_i$). Correlations between target observations, which our ConvCNP cannot model, significantly impact these measures, so we select a simpler measure: the mean standard deviation averaged over *target points* T, given a new hypothetical observation at $\mathbf{x}_i$ (set to the model's predicted mean at $\mathbf{x}_i$):

$$\tilde{\Gamma}(\mathbf{x}_i, t) = \frac{1}{N_T} \sum_{j=1}^{N_T} \sigma_\theta(\mathbf{x}_j | C(t)), \tag{5.5}$$

where σ_θ is the model's predicted standard deviation function.

For our target locations, we use a high-res grid over all of Germany. For the search locations, we choose a coarser grid, which is indicated in the respective Figs. 5.16 and 5.17. We choose a coarser grid because each search point $\mathbf{x}_i \in S$ requires a new prediction, so computational cost is of order $\mathcal{O}(|S|)$. As the computation of this acquisition function is very expensive, we can only average over 50 test tasks, which we choose randomly.

As we show in the case of Germany, the behaviour of σ_θ changes significantly during fine-tuning in the dense-station ($N_{stations} = 500$) regime (see Fig. 5.8). Given that Γ uses σ_θ as a foundation to make decisions, we expect station placement proposals within Germany to also change dramatically, indicating that Sim2Real is essential for active learning in dense-station regions.

We show that this is the case by contrasting the sim-only sensor placements with the Sim2Real sensor placements for $N_{stations} = 20$ and $N_{stations} = 500$ (with fine-tuning performed at the maximum $N_{times} = 10000$), in Figs. 5.16 and 5.17 respectively.

As expected, in the $N_{stations} = 20$ case, sensor placements are largely similar to the sim-only model, as Sim2Real has not significantly changed σ_θ. In contrast, for $N_{stations} = 500$, the sensor placements differ quite significantly, with the acquisition function being quite high in station-dense regions. The model has learned that measurements can be noisy and might have little predictive power in regions of high change, in turn increasing the value of stations in regions that already have many.

It would be interesting, in future work, to expand these experiments by running them at higher spatial resolutions, as the search points can have a significant impact on the placement locations, especially at such high station densities.

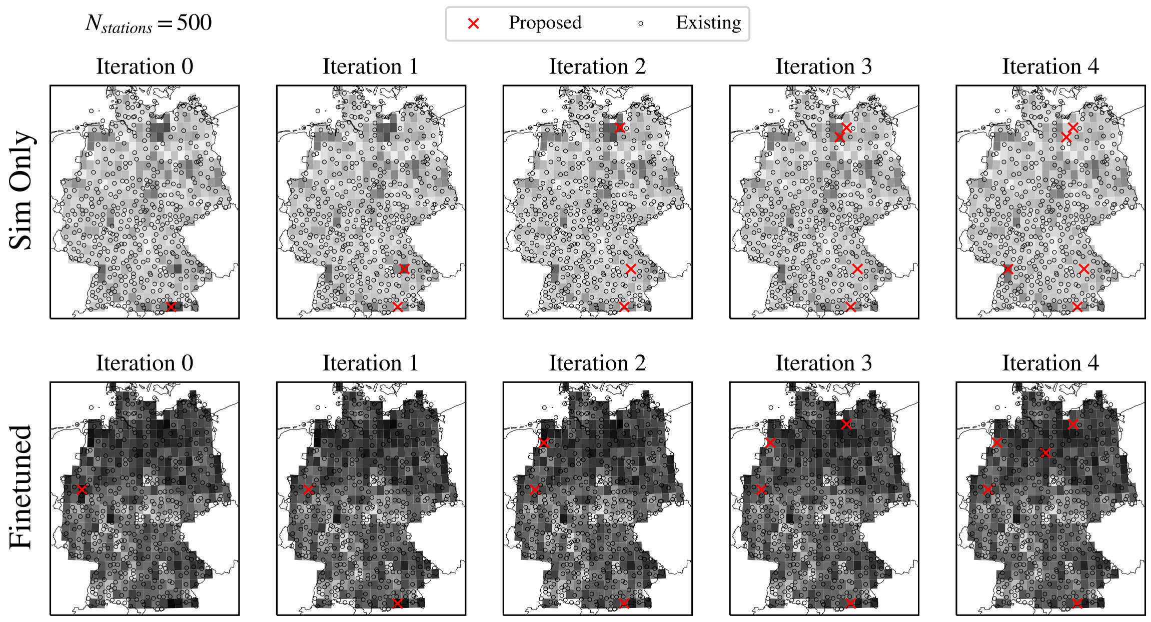


Figure 5.17: Fine-tuning the model on 500 stations significantly changes the auxiliary function across Germany, leading to vastly different proposed sensor placements.

5.13 Summary

Besides qualitatively and quantitatively showing the improvements in predictive means achieved by Sim2Real, we have shown that Sim2Real significantly improves the quantification of uncertainty associated with real measurements: the coarse simulator data are too smooth to accurately learn real uncertainties from. Higher aleatoric noise and short-scale weather phenomena significantly affect real measurements, and in domains in which the Sim2Real gap is large, this can have dramatic effects on downstream tasks.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.1.1 Sim2Real Is Effective for Finetuning ConvCNPs

As shown in both the synthetic GP-Regression experiment (Sec. 4) and the real-world temperature modelling experiment (Sec. 5.7), Sim2Real transfer is effective: Pre-training a ConvCNP on simulator data significantly reduces the amount of real data required when compared to training using only real data. It also significantly improves performance when compared to the simulator-only baseline, as long as the real data is sufficiently different.

6.1.2 Convolutional Filters Should be Tuned when Lengthscales Change

In both experiments, we find that globally tuning all parameters of the model beats tuning just the FiLM adapters, as long as the difference between simulator and real data involves a change in spatial lengthscales: in the temperature experiment, real data feature shorter lengthscales as some stations are more closely spaced than the simulator grid. In the synthetic experiment, we find that FiLM adapters are superior, *unless* the real data feature shorter lengthscales. To capture these higher-resolution features, the low-resolution convolutional filters, trained on lower-resolution simulator data, are insufficient.

6.1.3 Good Simulators cannot Replace Real Observations

In the temperature modelling experiment, we found that the model changes significantly during fine-tuning: from higher-frequency features to drastically different predicted uncertainties (Sec. 5.8). This also exposes the limitations of using simulators as the ground truth for data-driven weather modelling: even very sophisticated high-resolution simulations that include real observations via data assimilation, such as ERA5, should not be treated as a drop-in replacement for real data.

6.1.4 Data-Driven Models Need Different Sensor Placements

Whereas physics-based simulators can compute short-scale weather behaviour by approximating relevant physical laws on a discrete spatiotemporal grid, data-driven models need to learn this behaviour from data: If the model never receives training signals below a certain spatial lengthscale, we cannot expect it to learn accurate predictions in this region (see Sec. 5.10). This lengthscale is on the order of the shortest inter-station separation.

This imposes different requirements on sensor placements: physics-based simulators (roughly) profit from evenly spaced sensors, because prediction error (roughly) grows with distance from the sensor, as approximation errors compound away from measurements. For data-driven models, however, station locations should feature all lengthscales, even if it means very closely separated sensors. We see this in Sec. 5.4: a random selection of training stations during fine-tuning vastly outperforms choosing stations that are spread as far apart as possible. This could help inform future station placements: if data-driven models become the industry standard of weather modelling, their requirements should enter into the decision process of where new sensors should be placed.

In the future, manoeuvrable sensors could represent the best source of training data for ConvCNPs and other spatiotemporal data-driven models, as they can continuously sample in space, providing the model with signals at all lengthscales.

6.2 Future Work

Besides the proposals made throughout this thesis, there are many exciting directions for future work.

Artefacts The visual artefacts around context stations are a key limitation of our work (Sec. 5.10). Adding linear layers after the convolutional layers appears to help the model generalise spatially, reducing these artefacts (Fig. 5.14). In future work, this method should be explored further.

Correlations So far, we've modelled target observations as conditionally independent given the context set (i.e. predicting Gaussian probability distributions with diagonal covariance matrices). To improve predictions, obtain consistent samples from the posterior distribution, and unlock more acquisition functions for active learning, an interesting extension would be to model correlations. This could be done using Conv**G**NPs (Markou et al., 2021a), such as by Andersson et al. (2023), or using autoregressive evaluation of the ConvCNP, as demonstrated by Bruinsma et al. (2022).

Precipitation and Beyond Temperature is relatively easy to approximate well: surrounding temperatures and elevation can inform accurate predictions using simple methods. Elevation has a much less trivial relationship to precipitation (surrounding topography, not just the elevation at the target point matters significantly in this case), and one in which our method of injecting elevation data alongside the context encoding might thrive.

Resolution-Changes Throughout this thesis, we pre-train models at the highest internal resolution required for any *finetuning* tasks. For example, in the temperature downscaling experiments (Sec. 5), we pre-train the model at a spatial resolution that is significantly too high for the gridded simulator data, unnecessarily increasing computational cost. This enables us to simply plug-in real data (which requires this higher resolution) in the fine-tuning process. An interesting alternative could be to pre-train the model at a lower resolution and then inserting "translation" layers before fine-tuning. These layers could consist of convolutional filters at the lowest-lengthscale levels of the CNN that take in high-resolution data and are initialised in a way that simply blurs this data to the lower resolution. By adding some noise to the weights (to break the symmetry between parameters) and fine-tuning these layers, time and computational costs could be saved during pre-training.

Experiments Across Domains In this thesis, we have completed one simple synthetic experiment, with quite straightforward findings, and one complicated real-data experiment, which revealed some "sharp edges" for Sim2Real, such as the problem with discrete context and target points resulting in no signals beyond the smallest separation of points, which can manifest as visual artefacts (see Sec. 5.10).

It would be interesting to complete further real-world experiments, to see how far our findings generalise and if other "sharp edges" are revealed.

We expect that Sim2Real will become an important component of the rapidly developing future of data-driven weather and climate modelling, and beyond.

Bibliography

Tom R Andersson, Wessel P Bruinsma, Stratis Markou, Daniel C Jones, J Scott Hosking, James Requeima, Alejandro Coca-Castro, Anna Vaughan, Anna-Louise Ellis, Matthew Lazzara, et al. Active learning with convolutional gaussian neural processes for environmental sensor placement. *Environmental Data Science*, 2023.

Tom Robin Andersson. DeepSensor: A Python package for modelling environmental data with convolutional neural processes, July 2023. URL https://github.com/tom-andersson/deepsensor.

Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alex Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations. *arXiv e-prints*, pages arXiv–2306, 2023a.

Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alex Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations. *arXiv e-prints*, pages arXiv–2306, 2023b.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv e-prints*, pages arXiv–2211, 2022.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, pages 1–6, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wessel Bruinsma. Neuralprocesses. A framework for composing Neural Processes in Python., July 2023. URL https://github.com/wesselb/neuralprocesses.

Wessel Bruinsma, Stratis Markou, James Requeima, Andrew YK Foong, Tom Andersson, Anna Vaughan, Anthony Buonomo, Scott Hosking, and Richard E Turner. Autoregressive conditional neural processes. In *The Eleventh International Conference on Learning Representations*, 2022.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Kang Chen, Tao Han, Junchao Gong, Lei Bai, Fenghua Ling, Jing-Jia Luo, Xi Chen, Leiming Ma, Tianning Zhang, Rui Su, et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv e-prints*, pages arXiv–2304, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Adam M Clayton, Andrew C Lorenc, and Dale M Barker. Operational implementation of a hybrid ensemble/4d-var global data assimilation system at the met office. *Quarterly Journal of the Royal Meteorological Society*, 139(675):1445–1461, 2013.

PHILIPPE Courtier, J-N Thépaut, and Anthony Hollingsworth. A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387, 1994.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

David C Dowell, Curtis R Alexander, Eric P James, Stephen S Weygandt, Stanley G Benjamin, Geoffrey S Manikin, Benjamin T Blake, John M Brown, Joseph B Olson, Ming Hu, et al. The high-resolution rapid refresh (hrrr): An hourly updating convection-allowing forecast model. part i: Motivation and system description. *Weather and Forecasting*, 37(8):1371–1395, 2022.

Veronika Eyring, Sandrine Bony, Gerald A Meehl, Catherine A Senior, Bjorn Stevens, Ronald J Stouffer, and Karl E Taylor. Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958, 2016.

Tom G Farr, Paul A Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, et al. The shuttle radar topography mission. *Reviews of geophysics*, 45(2), 2007.

Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–. URL `https://github.com/Toblerity/Shapely`.

Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2019.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

Stephan Hoyer, Clark Fitzgerald, Joe Hamman, et al. xarray: v0.8.0, August 2016. URL http://dx.doi.org/10.5281/zenodo.59499.

Kelsey Jordahl, Joris Van den Bossche, Martin Fleischmann, Jacob Wasserman, James McBride, Jeffrey Gerard, Jeff Tratner, Matthew Perry, Adrian Garcia Badaracco, Carson Farmer, Geir Arne Hjelle, Alan D. Snow, Micah Cochran, Sean Gillies, Lucas Culbertson, Matt Bartos, Nick Eubank, maxalbert, Aleksey Bilogur, Sergio Rey, Christopher Ren, Dani Arribas-Bel, Leah Wasser, Levi John Wolf, Martin Journois, Joshua Wilson, Adam Greenhall, Chris Holdgraf, Filipe, and François Leblanc. geopandas/geopandas: v0.8.1, July 2020. URL https://doi.org/10.5281/zenodo.3946761.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv e-prints*, pages arXiv–2212, 2022.

Peter Lynch. The origins of computer weather prediction and climate modeling. *Journal of computational physics*, 227(7):3431–3444, 2008.

Douglas Maraun, Martin Widmann, José M Gutiérrez, Sven Kotlarski, Richard E Chandler, Elke Hertig, Joanna Wibig, Radan Huth, and Renate AI Wilcke. Value:

A framework to validate downscaling approaches for climate change studies. *Earth's Future*, 3(1):1–14, 2015.

Stratis Markou, James Requeima, Wessel Bruinsma, Anna Vaughan, and Richard E Turner. Practical conditional neural process via tractable dependent predictions. In *International Conference on Learning Representations*, 2021a.

Stratis Markou, James Requeima, Wessel Bruinsma, Anna Vaughan, and Richard E Turner. Practical conditional neural process via tractable dependent predictions. In *International Conference on Learning Representations*, 2021b.

Met Office. *Cartopy: a cartographic python library with a Matplotlib interface*. Exeter, Devon, 2010 - 2015. URL https://scitools.org.uk/cartopy.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv e-prints*, pages arXiv–2301, 2023.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

Anna Vaughan, Will Tebbutt, J Scott Hosking, and Richard E Turner. Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development*, 15(1):251–268, 2022.

Wetterdienst. Aktuelle stündliche stationsmessungen der lufttemperatur und luftfeuchte für deutschland., 2023.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv e-prints*, pages arXiv–2111, 2021.

Fuqing Zhang, Y Qiang Sun, Linus Magnusson, Roberto Buizza, Shian-Jiann Lin, Jan-Huey Chen, and Kerry Emanuel. What is the predictability limit of midlatitude weather? *Journal of the Atmospheric Sciences*, 76(4):1077–1091, 2019.

Appendix A: Adjusted Sim2Real Learning Rates for GP Regression

In the shrinking lengthscale experiment, we compute the relevant learning rate, starting at a basic learning rate $\alpha_0 = 1 \times 10^{-4}$ via:

$$\alpha(\ell^{real}, \text{tuner}) = \alpha_0 \times \begin{cases} 0.1, & \text{if } \ell^{real} = 0.2 \\ 0.2, & \text{if } \ell^{real} = 0.1, \\ 1, & \text{otherwise.} \end{cases} \times \begin{cases} 50, & \text{if tuner} = \text{FiLM}, \\ 1, & \text{otherwise.} \end{cases} . \quad (6.1)$$

In the other experiments, we found the differences between fine-tuning methods to be much bigger, and choosing a smaller learning rate for global fine-tuning (1×10^{-5} instead of 1×10^{-4}) to be sufficient to highlight performance differences.